NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

INVESTIGATION OF INCREASED FORWARD FLIGHT VELOCITIES OF HELICOPTERS USING HIGHER HARMONIC STALL CONTROL AND REVERSE VELOCITY ROTOR CONCEPT

by

Steven G. Van Riper

December 2003

Thesis Advisor:	E. R. Wood
Second Reader:	Raymond Shreeve

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2003	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Investigation of Increased Forward Flight Velocities of Helicopters Using Second Harmonic Control and Reverse Velocity Rotor Concept			**5. FUNDING NUMBERS**
6. AUTHOR(S) Steven G. Van Riper			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Described is the behavior of a rotorcraft equipped with Higher Harmonic Stall Control (HHSC) and a Reverse Velocity Rotor (RVR). Current rotorcraft are limited in forward flight speed by retreating blade stall and compressibility effects on the advancing blade. Stall occurs as the blade encounters increasingly severe reverse flow. HHSC enables conventional rotor systems to fly on the forward and aft sections of the rotor disk, greatly reducing reliance on the mixed flow regions defined by the advancing and retreating blades. Employment of the RVR allows lift generation while the rotor is experiencing reverse flow. A similar type of two per revolution (2/rev) input can be tailored to deliver maximum benefit to RVR equipped rotorcraft.

Modification of the Joint Army Navy Rotorcraft Analysis and Design (JANRAD) computer program allows 2/rev cyclic input, use of the RVR, and analysis using high fidelity graphical output to examine angle of attack, coefficient of lift, and air load. Computational results show performance gains in conventional helicopters and high speed flight potential for RVR equipped aircraft. The RVR is applied to the Joint Heavy Vertical Lift (JVHL) aircraft conceptual design for preliminary analysis. This conceptual design can be used as an indicator of the performance of a high speed RVR equipped aircraft.

14. SUBJECT TERMS Higher Harmonic Stall Control, Reverse Velocity Rotors, Heavy Lift Aircraft			**15. NUMBER OF PAGES** 163
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

INVESTIGATION OF INCREASED FORWARD FLIGHT VELOCITIES OF HELICOPTERS USING HIGHER HARMONIC STALL CONTROL AND REVERSE VELOCITY ROTOR CONCEPT

Steven G. Van Riper
Major, United States Army
B.S., Embry-Riddle Aeronautical University, 1991

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2003

Author: Steven G. Van Riper

Approved by: E. Roberts Wood
Thesis Advisor

Raymond Shreeve
Second Reader

Anthony J. Healey
Chairman, Department of Mechanical and Astronautical Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Described is the behavior of a rotorcraft equipped with Higher Harmonic Stall Control (HHSC) and a Reverse Velocity Rotor (RVR). Current rotorcraft are limited in forward flight speed by retreating blade stall and compressibility effects on the advancing blade. Stall occurs as the blade encounters increasingly severe reverse flow. HHSC enables conventional rotor systems to fly on the forward and aft sections of the rotor disk, greatly reducing reliance on the mixed flow regions defined by the advancing and retreating blades. Employment of the RVR allows lift generation while the rotor is experiencing reverse flow. A similar type of two per revolution (2/rev) input can be tailored to deliver maximum benefit to RVR equipped rotorcraft.

Modification of the Joint Army Navy Rotorcraft Analysis and Design (JANRAD) computer program allows 2/rev cyclic input, use of the RVR, and analysis using high fidelity graphical output to examine angle of attack, coefficient of lift, and air load. Computational results show performance gains in conventional helicopters and high speed flight potential for RVR equipped aircraft. The RVR is applied to the Joint Heavy Vertical Lift (JVHL) aircraft conceptual design for preliminary analysis. This conceptual design can be used as an indicator of the performance of a high speed RVR equipped aircraft.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank the Naval Postgraduate School Department of Aeronautics and Astronautics for their generous support. Great appreciation goes to all the professors in the department who gave me the education required to conduct this study. I am deeply indebted to my thesis advisor, Dr. E. Roberts Wood for his insightful guidance and enthusiasm. I would also like to thank Dr. Ray Shreeve for acting as the second reader of my thesis. Most importantly, I would like to thank my family for their support, not only during the conduct of the thesis but throughout all my graduate studies.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION FOR RESEARCH

The next generation of ship-based military vertical heavy lift rotorcraft must be capable of conducting high speed long-range combat assaults. It must be able to operate in all weather conditions, day or night, from prepared or unprepared surfaces. Operational requirements are shown in the figures below.

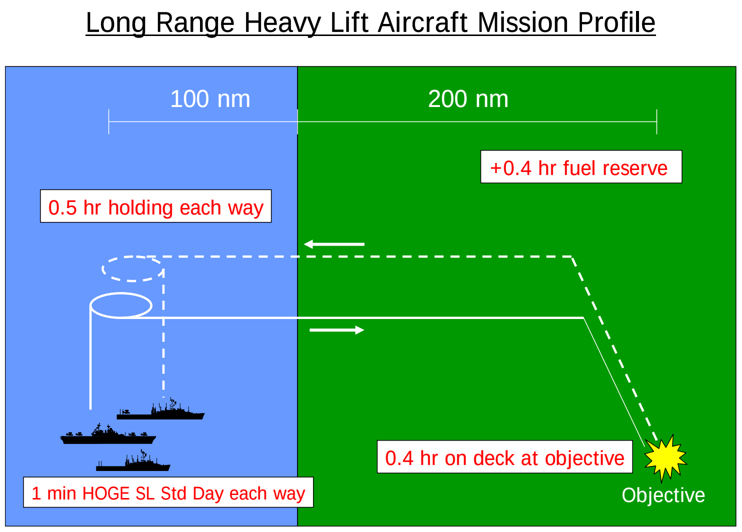

Long Range Heavy Lift Aircraft Mission Profile
100 nm
200 nm
+0.4 hr fuel reserve
0.5 hr holding each way
0.4 hr on deck at objective
1 min HOGE SL Std Day each way
Objective

Figure 1. Mission Profile.

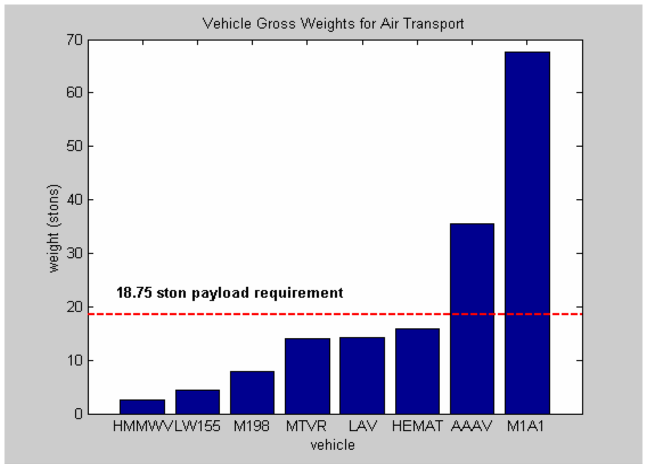


Figure 2. Payload Requirements.

These requirements are outside the envelope of any existing Vertical/Short Take-Off and Landing (VSTOL) aircraft to date. By the year 2020, the CH-46E will have been replaced by the MV-22A. But, the MV-22A does not have sufficient internal or external payload capacity to transport heavy equipment or large quantities of bulk logistics support. The CH-53E does not have sufficient internal load capacity to carry heavy equipment such as the Mk28/Mk928 7-ton truck needed to support USMC field artillery ashore. It does not have sufficient range with large external loads like the Light Armored Vehicle (LAV) to support the Operational Maneuver from the Sea (OMFTS) and Ship to Objective Maneuver (STOM) operations up to 200 NM inland envisioned in the future. Although a modified CH-53X has been proposed, no manufacturing plan has been formalized and current plans call for the phase out of existing CH-53Es by 2025. Clearly, a new aircraft must be developed to meet future requirements.

B. HIGHER HARMONIC STALL CONTROL AND REVERSE VELOCITY ROTORS

Two of the most compelling technologies that might be used to meet these operational requirements are Higher Harmonic Stall Control (HHSC) and Reverse Velocity Rotors (RVR). Both of these technologies have been studied in great detail but have never enjoyed widespread acceptance within the helicopter industry. An aircraft using HHSC and the RVR system would retain all the characteristics of a conventional helicopter at low airspeeds but would also be capable of high speed flight (in excess of 200 knots).

HHSC differs from Higher Harmonic Control (HHC) in the following manner. HHC is an active control concept whose principal objective has been to reduce helicopter airframe vibrations. More recently it has been applied to improve helicopter performance as well. For HHC principle frequencies of interest are $(n-1)\Omega$, $n\Omega$, and $(n+1)\Omega$ in the rotating system and $n\Omega$ in the fixed fuselage system. Here, n, is the number of blades and Ω is the rotor rotational speed. [Wood et. al, 1985] In contrast to HHC, HHSC relates usually to 2Ω or second harmonic frequency. [Arcidiacono, 1961] More recently it has been extended to 3Ω frequency as well. The objective of HHSC when applied to the rotor in high-speed flight is to smooth out the distribution of lift and eliminate stall over the entire rotor disk.

A preliminary design of an RVR based vertical heavy lift aircraft was developed but analysis and verification of rotor performance was nearly impossible since traditional design methodology and performance calculations worked well with the rotorcraft only up to 170 knots. The planned cruise speed of the RVR based aircraft was 300 knots. After several attempts to constrain our design to work with traditional rotor design tools it was determined the best course of action would be to develop a new way to investigate the behavior of the rotor system in which both RVR and HHSC technology would be applied.

C. JANRAD 2P_RVR

Due to the unique nature of RVR and HHSC the original Joint Army Navy Rotorcraft Analysis and Design (JANRAD) program was modified to allow for HHSC, use of RVRs, and several other enhancements. The core pitch-thrust moment iteration technology that forms the core of JANRAD was unchanged.

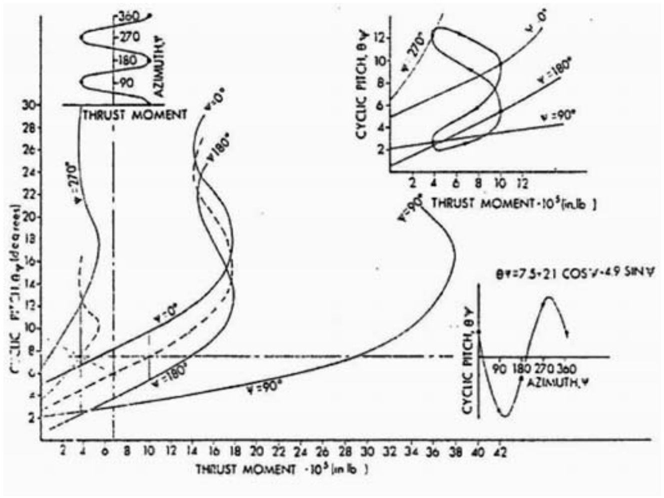


Figure 3. Graphical Depiction of Pitch-Thrust Moment Iteration. [From: Gerstenberger and Wood, 1963]

A complete description of the original program, assumptions, and operating constraints can be found in theses by Nicholson [Nicholson, 1993] and Eccles. [Eccles, 1995] Sections two and three discuss the employment and effects of using the HHSC and RVR systems. These chapters will provide a working knowledge of how each system works by outlining program modifications, assumptions, and actual JANRAD 2P_RVR output. UH-60 aircraft data will be used to further illustrate applicability to current helicopter design. JANRAD 2P_RVR was used for the analysis of the Joint Vertical Heavy Lift Aircraft (JVHL) outlined in section four. Complete user instructions for JANRAD 2P_RVR are provided in section five.

The study concludes with a brief list of recommendations for improvement and suggestions for further study. A complete listing of all the modules required to run the JANRAD 2P_RVR program is given in the appendices.

II. HIGHER HARMONIC STALL CONTROL

A. INTRODUCTION

Since the early 1960s researchers have pursued the idea of incorporating two per revolution (2/rev or 2P) control inputs to rotor systems. Studies ranged from increasing forward fight speed to the reduction of aircraft vibrations. Computer controlled Higher Harmonic Control (HHC), 3P, 4P and 5P, showed significant vibration reduction and was the subject of a detailed flight test program conducted by Hughes Helicopters, Inc. under a NASA/Army contract. [Wood et. al, 1985] For the study of this thesis the emphasis will be on performance airspeed increases and controlling the rotor stall pattern in which 2P and 3P Higher Harmonic Stall Control (HHSC) will be applied.

B. BLADE PITCH MOTION AND THE SWASHPLATE

Blade pitch motion comes from two sources, namely commanded collective and cyclic pitch input from the helicopter control system and elastic deformations (twist) of the blade and control system. This section will focus on the cyclic commanded input. Elastic deformations are neglected at this level of study.

The pitch time history of the rotor blade can be represented by a Fourier series as follows:

$$\theta\ (\psi) = \theta_0 + \theta_{1c}\cos(\psi) + \theta_{1s}\sin(\psi) + ... + \theta_{nc}\cos(n\psi) + \theta_{ns}\sin(n\psi) + ...$$

where θ is the blade's feathering angle, ψ is the blade azimuth position, and $\psi = \Omega t$. Conventional control inputs produced by the swashplate consist of collective pitch, θ_0, and the first harmonics of the Fourier series: the lateral cyclic θ_{1c} and the longitudinal cyclic θ_{1s}. A conventional blade pitch (feathering) motion can be described by the Fourier series

$$\theta\ (\psi) = \theta_0 + \theta_{1c}\cos(\psi) + \theta_{1s}\sin(\psi) = \theta_0 + \theta_{1c}\cos(\Omega t) + \theta_{1s}\sin(\Omega t)$$

This one per revolution control (1/rev or 1P) formed the core of JANRAD's original trim routine. The θ_{1c} term controls the lateral orientation of the rotor disk and θ_{1s} term controls the longitudinal orientation. In the above equation, $t = 0$ has been taken with the blade in the trailing or aft ($\psi = 0$) position. Remember that for a rotor with a centrally located flap hinge there is an exact 90 degree force/displacement phase lag. In the case of pure θ_{1s} (sine) longitudinal cyclic motion, the minimum aerodynamic flapping motion occurs at ψ = 90 degrees, where the minimum flapping displacement occurs 90 degrees later at ψ = 180 degrees; the maximum aerodynamic flapping moment occurs at ψ = 270 degrees, and the maximum flapping displacement occurs 90 degrees later, at ψ = 360 degrees.

The swashplate is the key to affecting the pitch control of the rotor blades. The swashplate has rotating and nonrotating disks concentric with the rotor shaft. Both disks slide up and down in response to collective inputs or they can be tilted to an arbitrary orientation in response to cyclic inputs made by the pilot. The pilot cyclically changes the pitch of the blades about the swashplate. HHSC inputs can be incorporated into existing helicopters by modification of the swashplate. HHSC inputs would be transmitted to the rotor using a stationary outer member with a track attached to a conventional swashplate assembly. This differs dramatically from HHC systems which rely on computer controlled actuators connected to the swashplate to alter the rotor pitch. Figure 4 compares a HHC system with a proposed HHSC mechanism.

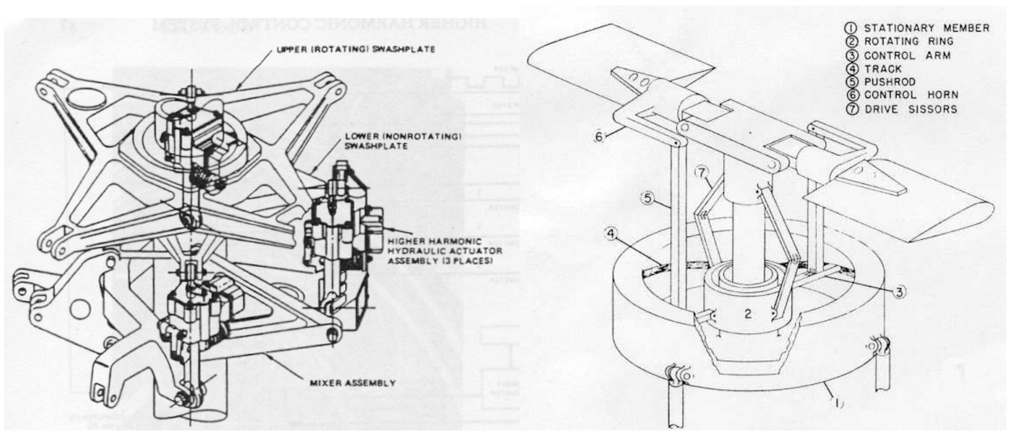

UPPER (ROTATING) SWASHPLATE
LOWER (NONROTATING) SWASHPLATE
HIGHER HARMONIC HYDRAULIC ACTUATOR ASSEMBLY (3 PLACES)
MIXER ASSEMBLY
① STATIONARY MEMBER
② ROTATING RING
③ CONTROL ARM
④ TRACK
⑤ PUSHROD
⑥ CONTROL HORN
⑦ DRIVE SISSORS

Figure 4. HHC Actuator System [From: Wood et. al, 1985] Compared to a HHSC Swashplate and Ring System. [From: Arcidiacono, 1961]

C. EXPERIMENTAL ANALYSIS USING JANRAD 2P_RVR

Since research by Arcidiacono has suggested a significant increase in maximum forward speed of a helicopter could be obtained through the use of HHSC control of rotor pitch the first modifications to JANRAD were made to the Fourier series that defines cyclic pitch. First iterations of JANRAD 2P_RVR used the following equation for θ given by

$$\theta\,(\psi) = \theta_0 + \theta_{1c}\cos(\psi) + \theta_{1s}\sin(\psi) + \theta_{2c}\cos(2\psi) + \theta_{2s}\sin(2\psi) + \theta_{3s}\sin(3\psi + \theta_{offset})$$

This equation incorporates lateral and longitudinal 2P signals and longitudinal 3P signals plus a phase, θ_{offset}. The 2P and 3P θ coefficients are user defined fixed values while the 1P coefficients are determined by JANRAD based on first harmonic trimming rules. Since the objective of our study was to maximize performance it was found that the best equation for θ was found to be

$$\theta\,(\psi) = \theta_0 + \theta_{1c}\cos(\psi) + \theta_{1s}\sin(\psi) + \theta_{2c}\cos(2\psi) + \theta_{3s}\sin(3\psi + \theta_{offset})$$

The 2P sine term was eliminated because it was causing increased rotor stall on the left and right side of the rotor disk, in the exact location were we wished to reduce rotor stall. The 3P sine term produced striking results and was retained for experimental value, but in the majority of cases this term was assigned a value of zero.

D. TEST USING PROUTY'S HELICOPTER

As stated earlier, the primary reason for using HHSC (2P) is to improve the lifting efficiency of the rotor by modifying the stall pattern. Before attempting to analyze 2P behavior with JANRAD 2P_RVR, conventional first harmonic inputs were verified by comparing basic output with known results. Prouty's sample helicopter was used as the baseline for comparison. To determine if JANRAD 2P_RVR was producing correct output, rotor disk angle of attack contour plots were compared to known solutions. These plots were chosen because the stall patterns are directly linked to the angle of attack distribution. All 2P and 3P terms were set to zero for the test. Figure 5 shows the calculated first harmonic cyclic pitch of the Prouty example helicopter.

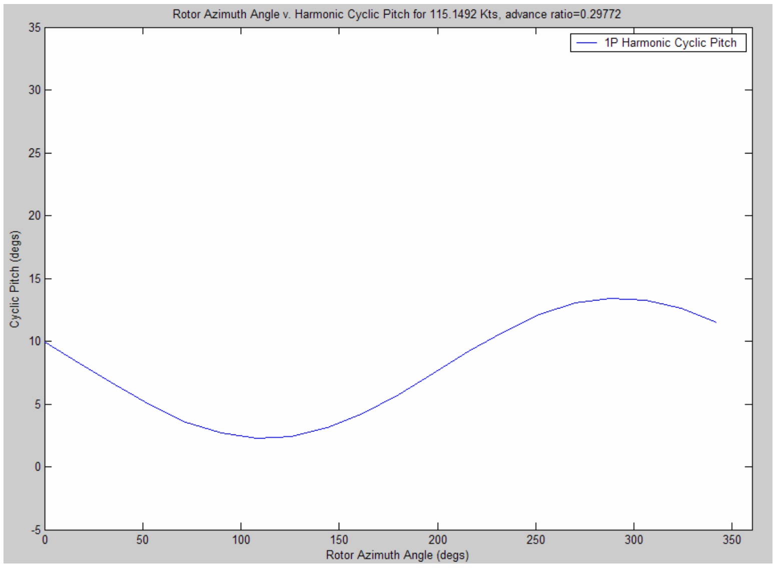


Figure 5. Rotor Azimuth versus Harmonic Cyclic Pitch for the Prouty Example Helicopter.

Figure 6 compares the angle of attack contour plot from Prouty's text to JANRAD 1P2P3P output.

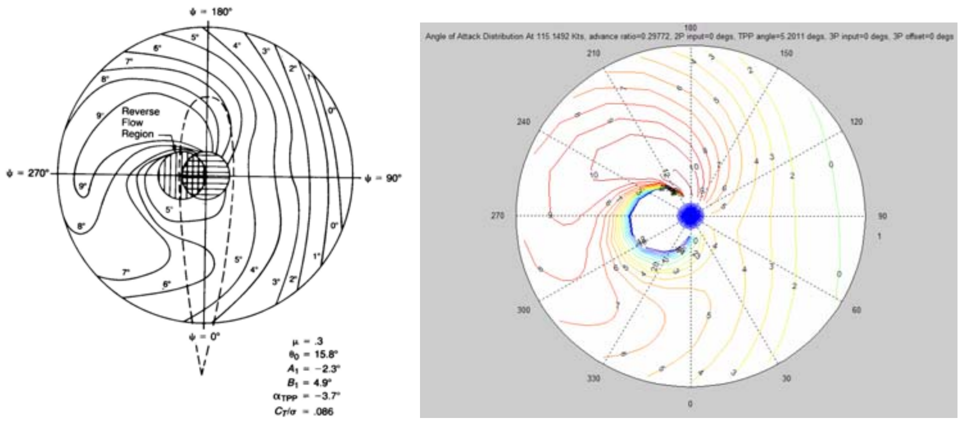


Figure 6. Prouty's Example Helicopter Contour Plot (Left) Compared to JANRAD 2P_RVR Output (Right) Using Prouty's Example Helicopter.

The similarities between the two plots indicate JANRAD 2P_RVR output is stable and in close agreement with Prouty's numerical results. Next, 2P and 3P signals were introduced. Figures 7 and 8 show the result for a 2P signal and 3P signal, respectively.

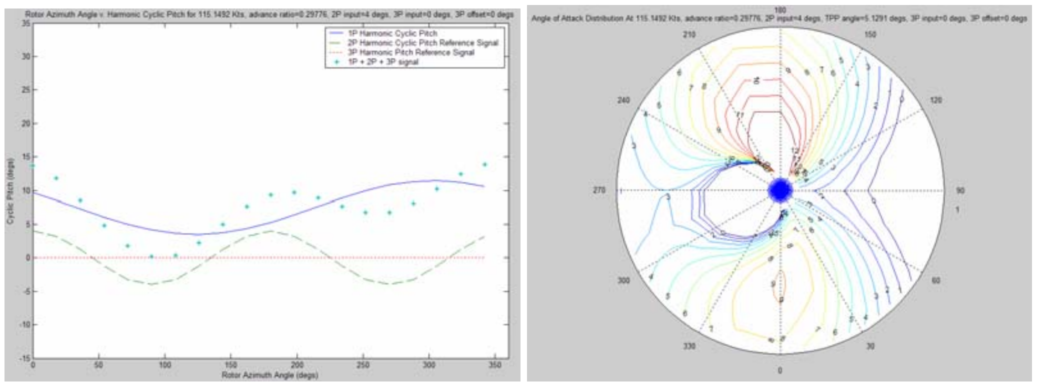


Figure 7. 2P Signal Effects on Cyclic Pitch (Left) and Angle of Attack (Right).

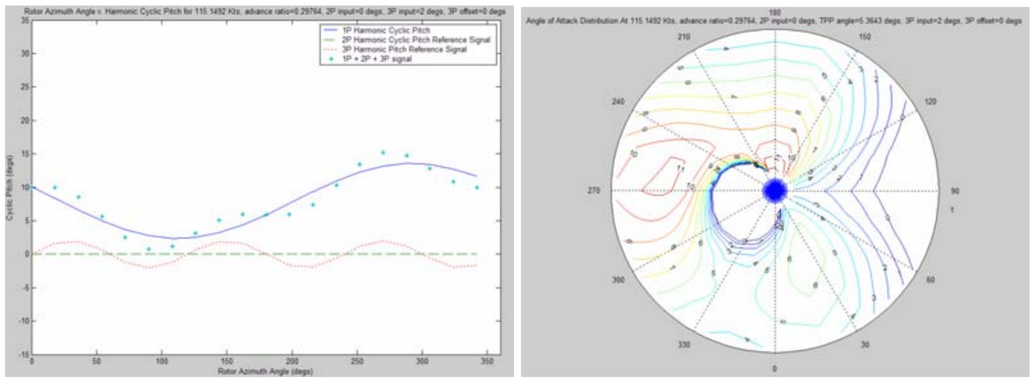


Figure 8. 3P Signal Effects on Cyclic Pitch (Left) and Angle of Attack (Right).

Three dimensional lift distribution plots were used to further verify the effects of these higher harmonic signals. The plots are compared to conventional lift distributions in Figures 9 and 10.

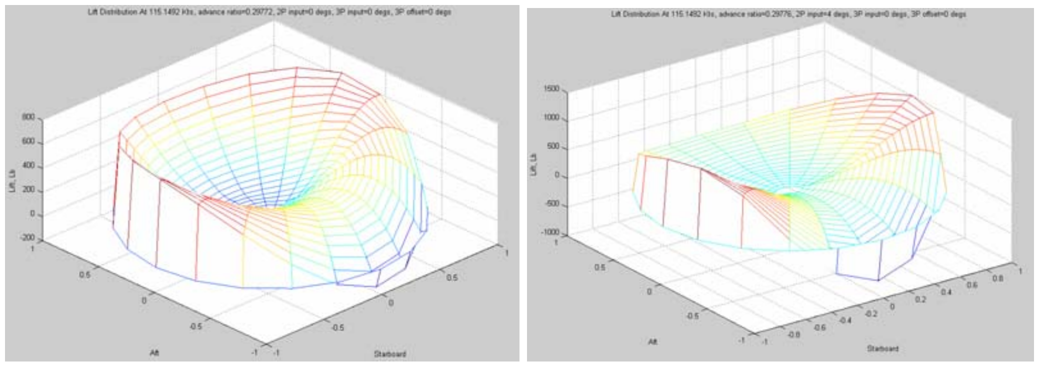


Figure 9. Conventional (1P) Lift Distribution (Left) Compared to 2P Lift Distribution.

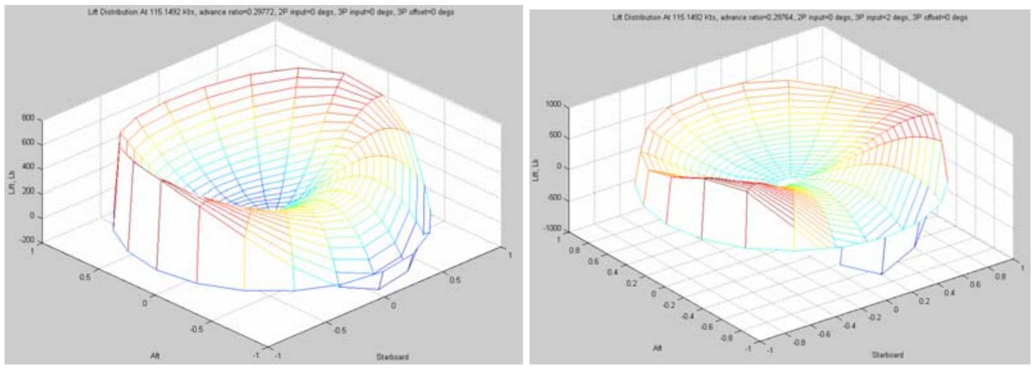


Figure 10. Conventional (1P) Lift Distribution (Left) Compared to 3P Lift Distribution.

These plots clearly show how the 2P signal moves the aerodynamic forces to the forward and aft sections of the rotor disk. By doing this the helicopter can 'fly' on the forward and aft sections of the disk and therefore not rely as heavily on the left and right sections. The rotor is no longer trying to provide propulsive and lifting force from the unstable left and right rotor regions. This translates into the ability to gain forward velocity. As was mentioned earlier, the 3P signal was ineffective in showing any performance gains but was retained for experimental value.

E. COMPARISON TO ARCIDIACONO

One additional test was run to verify JANRAD 2P_RVR's integrity. Thrust distribution and angle and angle of attack distribution were compared to ideal thrust and angle of attack distributions derived by Arcidiacono. [Arcidiacono, 1961] Figures 11 and 13 show Arcidiacono's original plots and his attempts to meet the optimum distribution. Note that Arcidiacono's sign conventional is opposite to the sign conventional used throughout this study.

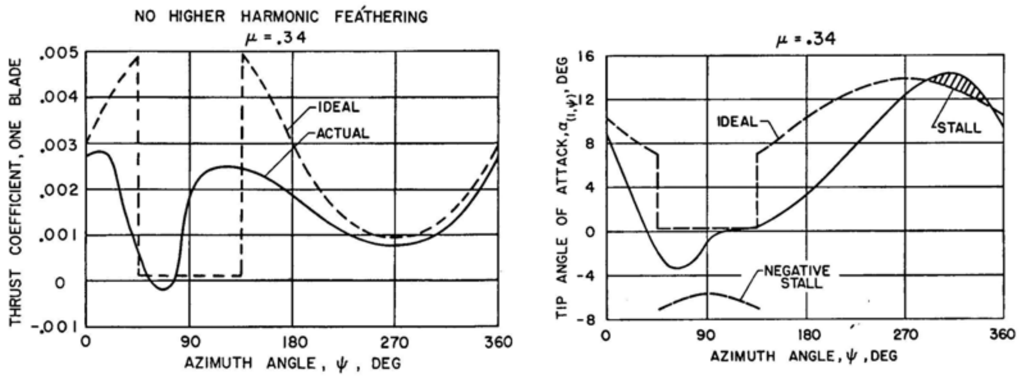


Figure 11. Arcidiacono's Results with No 2P.

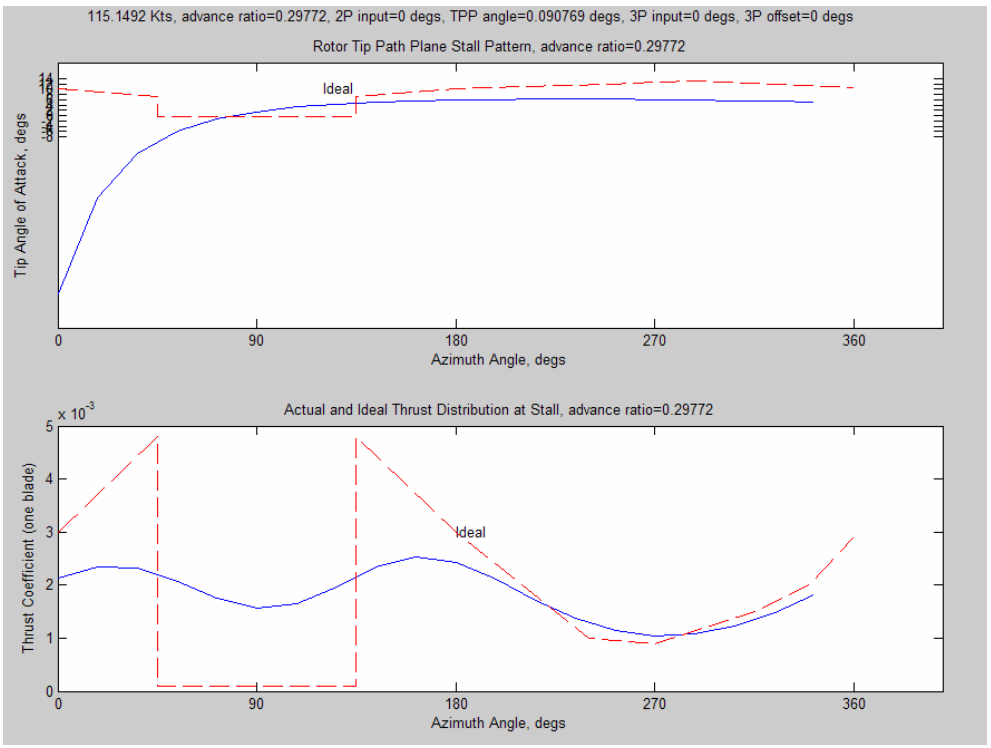


Figure 12. Results Using JANRAD 2P_RVR with No 2P.

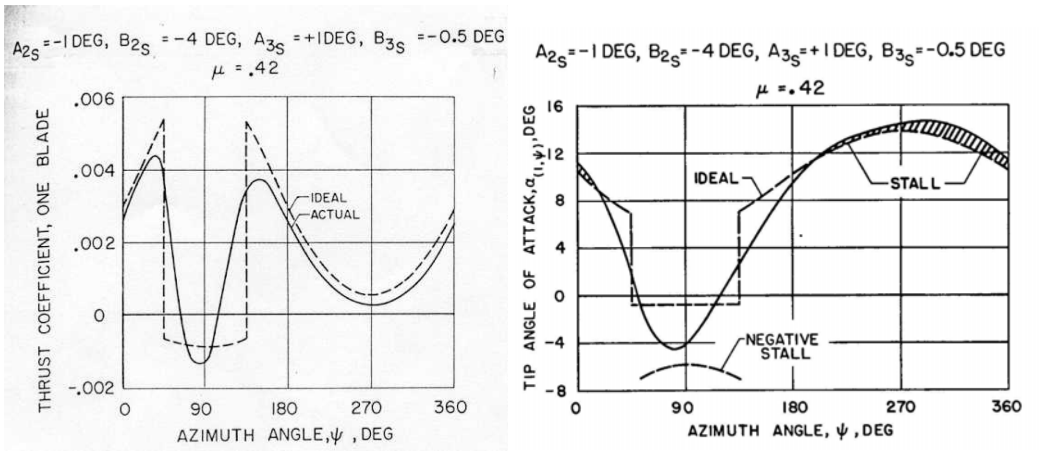


Figure 13. Arcidiacono's Results with 2P and 3P. [From: Arcidiacono, 1961]

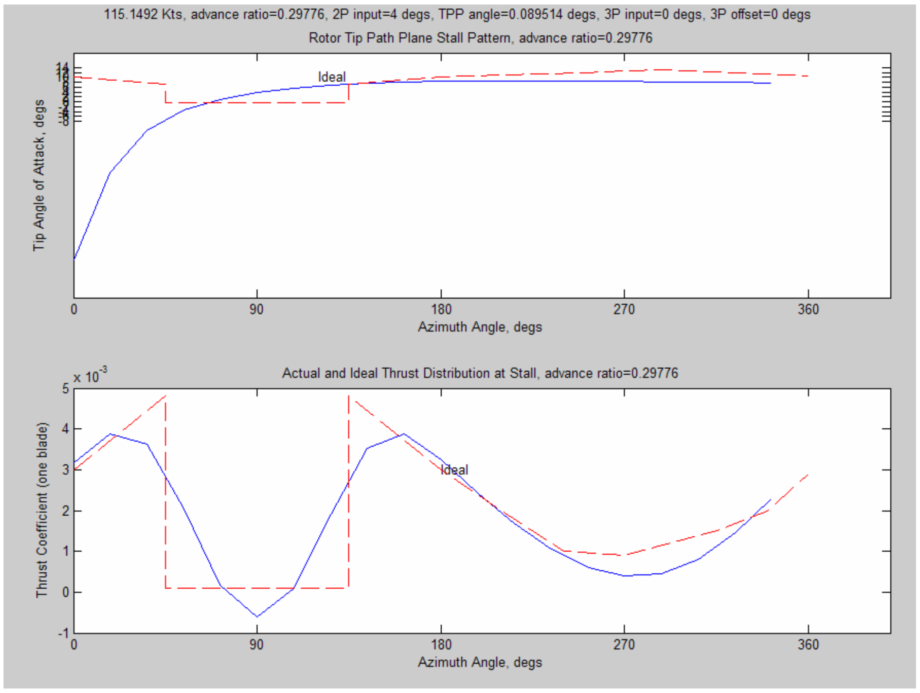


Figure 14. Results Using JANRAD 2P_RVR with 2P.

Figure 14 shows that JANRAD results were favorable in terms of thrust coefficient. Angle of attack distribution was expected to be less favorable due to the constraints imposed by JANRAD for tip losses. Overall, JANRAD 2P_RVR validated itself for use with 2P inputs.

F. FORWARD FLIGHT VELOCITY INCREASES

Previous versions of JANRAD had been validated for use with the UH-60A; therefore it was selected as the helicopter to test for potential speed increases. [Eccles, 1995] All UH-60A parameters were loaded into JANRAD 2P_RVR. Sikorsky's proprietary airfoil, the SC1095R8, was used as the rotor blade. Testing was confined to conventional helicopter configurations only.

In short, an increase in maximum flight speed of 14 knots was achieved using a four degree fixed 2P input. This is just short of a ten percent improvement. No significant difference in maximum forward speed was observed with 2P inputs greater than 4 degrees. This increase was observed under the 1.5 percent iterative deviation rule imposed by JANRAD 2P_RVR. That is, collective and cyclic pitch settings are adjusted until the thrust vector is within 1.5 percent of the previous iteration. This stringent requirement ensures realistic simulation within the trim module.

The maximum airspeed without any 2P input was 159 knots. The rotor is under a great deal of stress to produce all the required forces for flight. Figures 15 through 18 depict the rotor behavior under these conditions. Any increase in speed caused JANRAD 2P_RVR to diverge and return to the user a 'WILL NOT TRIM' message.

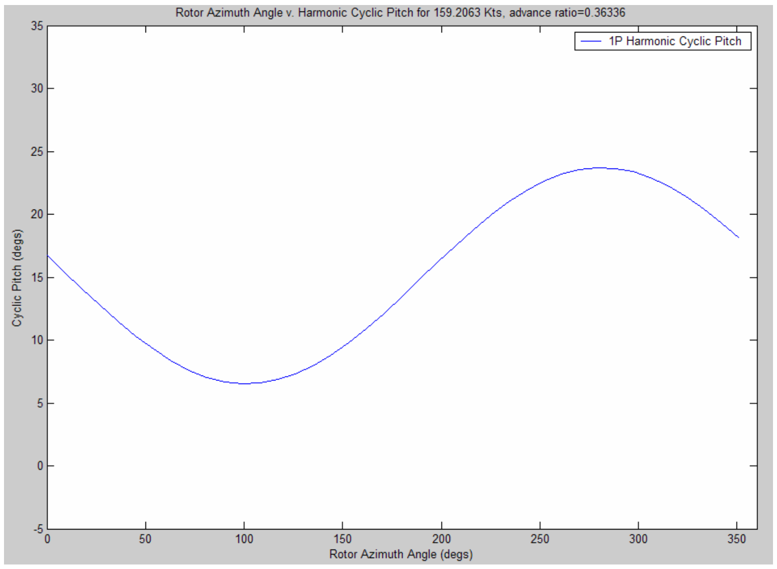


Figure 15. UH-60A Cyclic Pitch at 159 Knots.

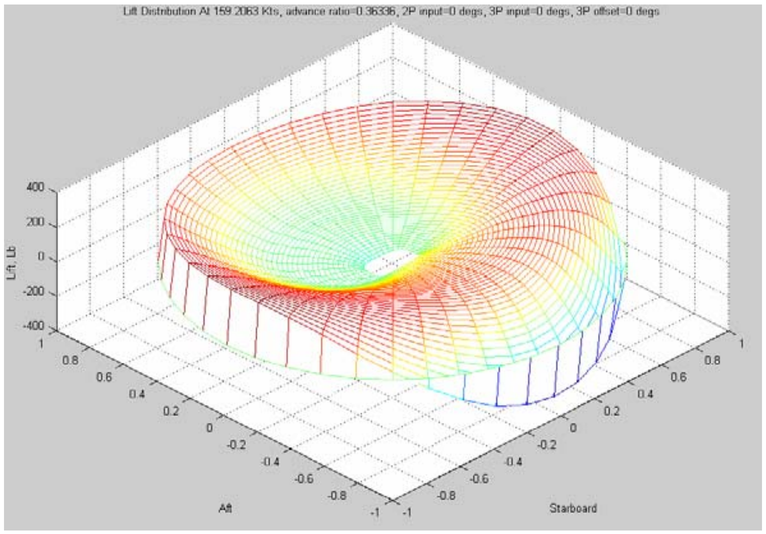


Figure 16. UH-60A Lift Distribution at 159 Knots.

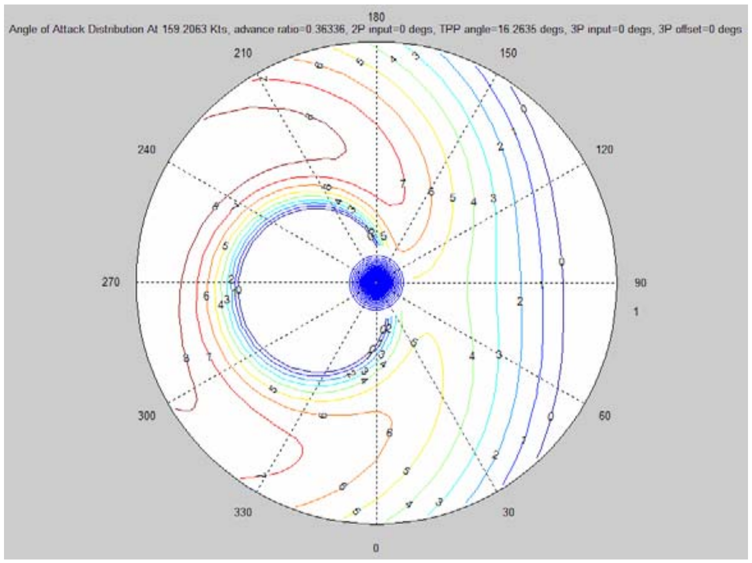


Figure 17. UH-60A Angle of Attack Distribution at 159 Knots.

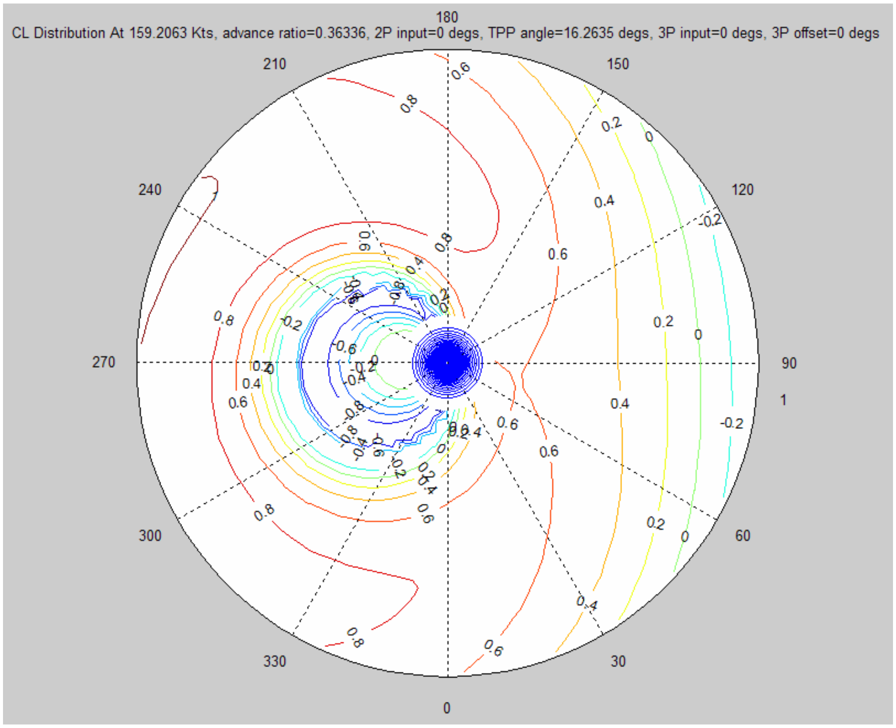


Figure 18. UH-60A C_L Distribution at 159 Knots.

The maximum airspeed with the four degree 2P input was 173 knots. The familiar 2P pattern is clearly evident in Figure 21. The other figures show results similar in appearance to the plots derived from Prouty's example helicopter. Any increase in speed caused JANRAD 2P_RVR to diverge and return to the user a 'WILL NOT TRIM' message.

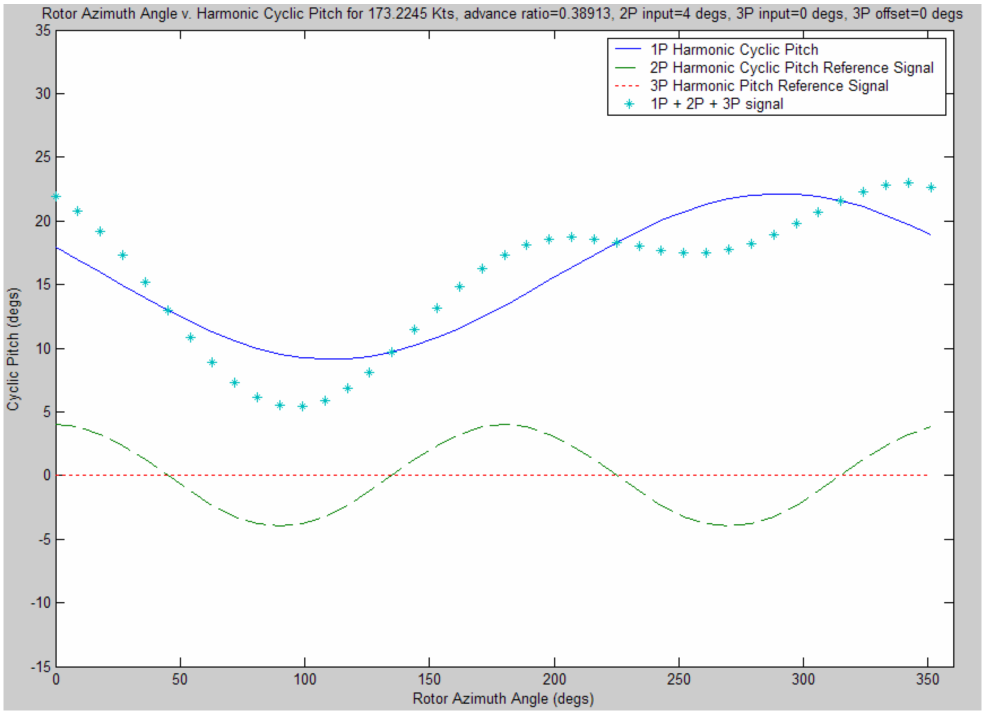


Figure 19. UH-60A 2P Cyclic Pitch at 173 Knots.

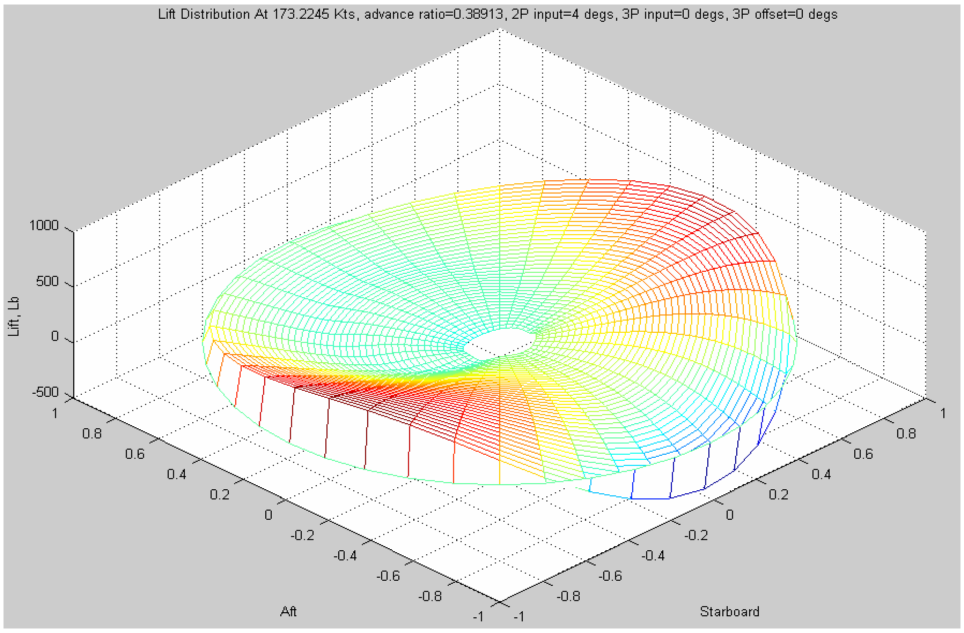


Figure 20. UH-60A 2P Lift Distribution at 173 Knots.

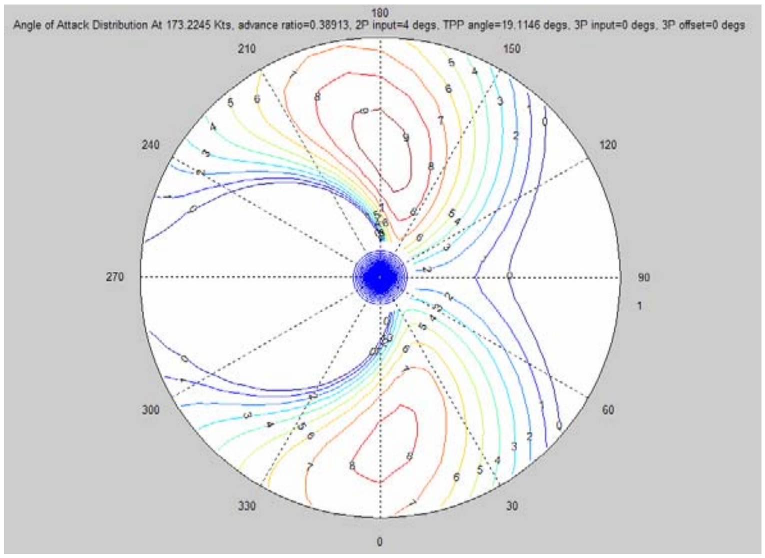


Figure 21. UH-60A 2P Angle of Attack Distribution at 173 Knots.

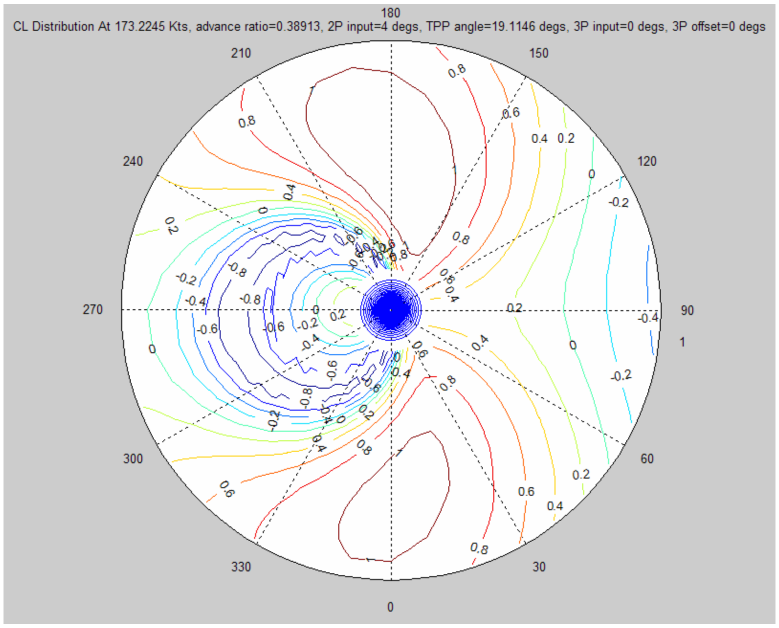


Figure 22. UH-60A 2P C_L Distribution at 173 Knots.

Although not shown, testing was also carried out with compound helicopters. Use of 2P systems with compound helicopters equipped with auxiliary thrust devices performed very similarly to conventional 2P equipped helicopters while the rotor is providing forward thrusting power. As the auxiliary thrust system begins to provide the majority of propulsive force at airspeeds above 180 knots, if no power was supplied to the rotor, the disk would assume an aft tilting tip path plane and rotor drag would increase significantly. This situation is avoided completely in actual compound helicopters in high-speed flight by providing a relatively small amount of power to stabilize the rotor, while keeping the tip path plane at a near zero angle of attack.

The application of second harmonic control is not a new idea for improving the speed performance of helicopters. This thesis has verified with JANRAD 2P_RVR that a 2P rotor system does trim at airspeeds up to 10% higher than conventional 1P rotor systems. The next section explores the use of Reverse Velocity Rotors coupled with a HHSC control signal.

III. EMPLOYMENT OF REVERSE VELOCITY ROTORS

Current helicopters are limited in their speed by three effects, namely stalling of the tip of the retreating blade, loss of lift due to reverse flow over the inboard portion of the retreating blade, and compressibility on the tip of the advancing blade. [Ewans, 1973] The severity of these effects is a function of the rotational speed of the rotor blades and the forward velocity of the aircraft. Note that if we treat retreating blade stall by slowing the rotor we also reduce compressibility effect on the advancing blade. Several studies have been conducted to determine the best way to combat retreating blade stall. Everything from exotic blade shapes, segmented rotors [Zientek, 2001] and forced air/circulation control have been attempted.

A. REVERSE VELOCITY ROTORS

A relatively new and promising concept recently introduced is called the Reverse Velocity Rotor (RVR). RVR systems are a recent development, invented by Harold Lemont to address the aerodynamic issues contributing to retreating blade stall and advancing blade compressibility phenomena, the primary factors that currently limit helicopter maximum forward flight speeds. Instead of eliminating the reverse flow that occurs on the retreating side of the blade the RVR is actually designed to operate in the reverse flow region. With conventional rotor blades, this would completely eliminate the potential to generate lift. In fact, a 'reversible' airfoil section with a rounded trailing edge is employed to give reasonable lift and drag characteristics in the reverse flow. Reverse velocity rotor systems are built around double-ended airfoils as shown in Figure 23.

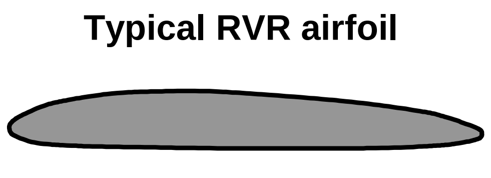


Figure 23. RVR Cross Section. [From: Ashby, 2002]

Reverse velocity rotor systems represent a revolutionary approach to high-speed Vertical Take Off and Landing (VTOL) configurations. Application of this concept makes possible rotorcraft designs that are capable of attaining speeds significantly greater than conventional or even compound helicopters. Initial analysis shows cruise speeds in excess of 300 knots are possible with RVR systems. Sikorsky Aircraft Corporation has completed a study that shows future Runway Independent Aircraft (RIA) equipped with RVR systems could attain cruise speeds over 300 knots, double the cruise speed of conventional helicopters. [Ashby, 2002] This airfoil design minimizes the impact of retreating blade apparent velocity reduction caused by summing the blades rotational velocity and the aircraft's forward flight velocity as illustrated in Figure 24 below.

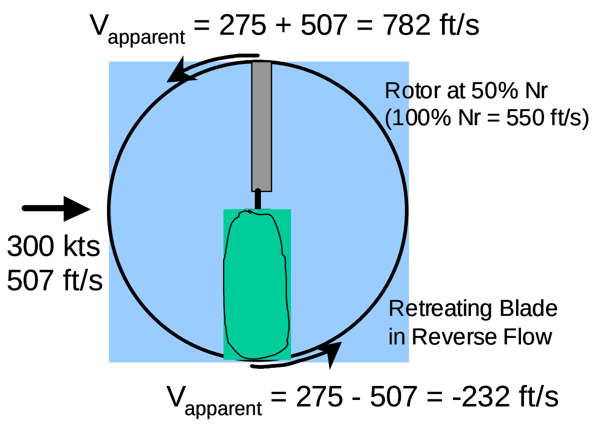


Figure 24. Rotor Disk Apparent Velocities. [From: Ashby, 2002]

As might be expected the RVR system does require somewhat more power to hover and fly at low airspeeds. For this reason the best application for RVR systems is for rotorcraft that will spend the majority of time in cruise configuration, i.e. civilian or military transport. Aircraft that will spend the majority of their flight time in low speed flight or hovering, i.e. rescue aircraft, should continue to employ conventional rotor blades. Figure 25 shows predicted power requirements.

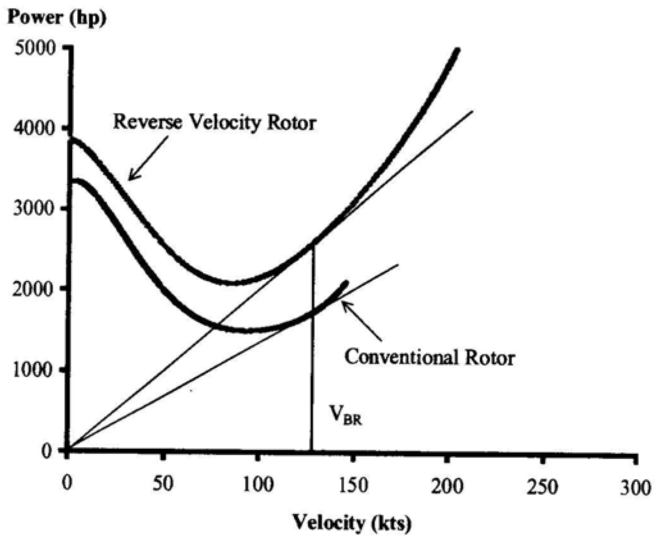


Figure 25. Power Requirements versus Velocity. [From: Zientek, 2001]

B. WIND TUNNEL TEST

Fairchild Republic Division conducted wind tunnel tests at NASA Ames in 1973 of a 1/7 scale 12% thickness double ended airfoil at equivalent airspeeds of over 310 kts. [Ewans, 1973] The test rig is shown in Figure 26.

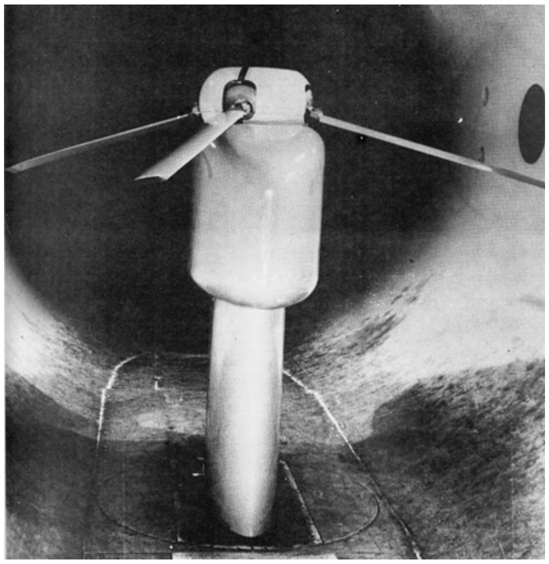


Figure 26. Installation of RVR Test Rig in the NASA Ames 12 Foot Wind Tunnel. [From: Ashby, 2002]

Testing was considered a success although some large unbalanced forces were noted at the edge of the test envelope. Data collection focused on measuring rotor performance, particularly maximum lift and effective lift/drag ratio over the full envelope. [Ewans, 1975] The most important data for this study are given in Figures 27 and 28. From these plots data tables were generated which correlated Mach number, angle of attack, and C_L or C_D. These tables were inserted into a MATLAB function which used a two-dimensional interpolation to provide the JANRAD 2P_RVR thrust moment and drag moment routines with the appropriate C_L and C_D values.

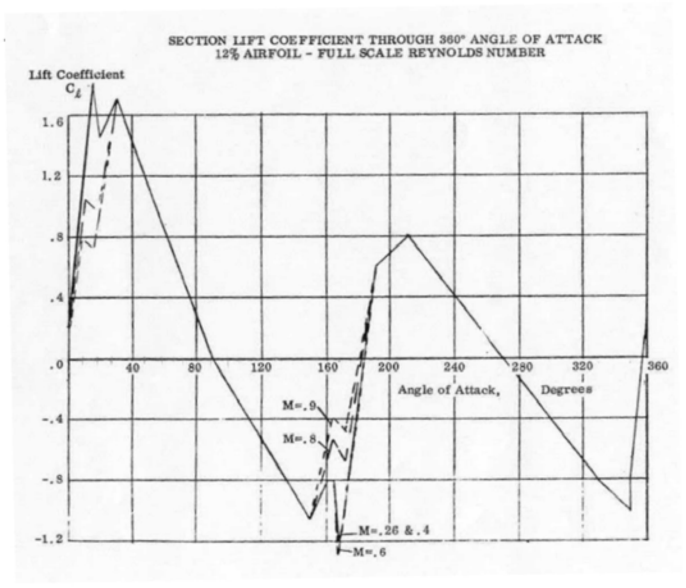


Figure 27. RVR C_L versus Angle of Attack. [From: Ewans, 1975]

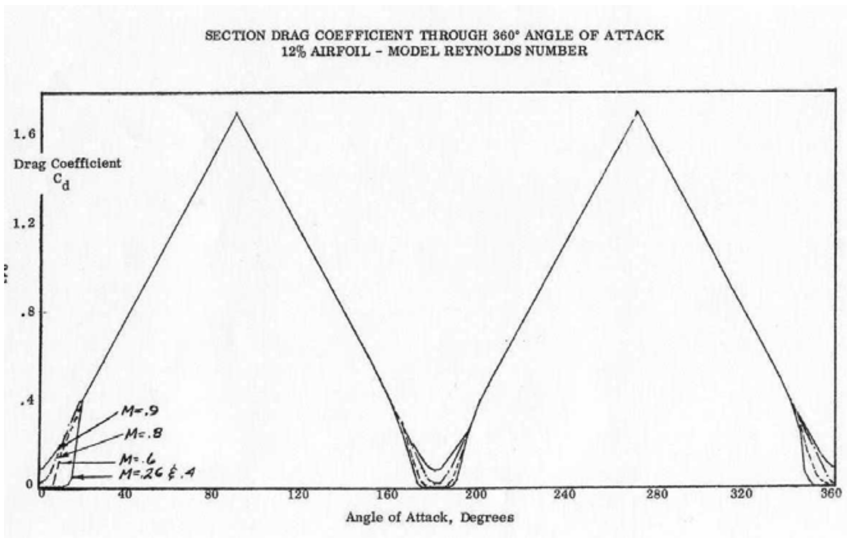


Figure 28. RVR C_D versus Angle of Attack. [From: Ewans, 1975]

C. EXPERIMENTAL ANALYSIS USING JANRAD 2P_RVR

JANRAD, in its original form, was not set up to calculate trim parameters for a rotorcraft traveling at 300 knots. Several new assumptions had to be made to make JANRAD 2P_RVR compatible with high speed RVR systems.

1. Advance Ratio

Before outlining the assumptions, the advance ratio parameter must be defined. The advance ratio is the ratio of the rotor's horizontal speed to its tip speed. The tip speed ratio, μ, is defined as:

$$\mu = V_{\infty} \cos\alpha / \Omega R$$

where

V_{∞} = forward airspeed

α = tip path plane angle

Ω = rotational velocity in radians per second

R = rotor radius

With a conventional rotor system as μ exceeds 0.4, collective and cyclic pitch values were unable to satisfy rotor lift, rotor drag, or resultant thrust vector requirements. The system falls out of 'trim' with respect to pitching, rolling and yawing moments. Even after applying second or HHSC the program was unable to trim above advance ratios of .5. This is not surprising when we recognize that at μ = 0.5, one half of the retreating blade (ψ = 270 degrees) is in reverse flow. In contrast, RVR systems operate in an environment with advance ratios ranging between 1.0 and 2.0. This fivefold increase in advance ratio in RVR aircraft over aircraft equipped with conventional rotors illustrates the significant differences between the two systems.

2. RVR Assumptions and Requirements

Several assumptions were made to enable computational trimming of an RVR system using the traditional JANRAD trimming program. For high-speed rotorcraft flight ($\mu \geq 0.5$) JANRAD 2P_RVR only allows RVR trimming on compound helicopters where a wing provides the majority of vertical lift at cruise speeds. Proper employment of a wing can reduce rotor loads up to 80 percent. The reduction in rotor loading eliminates blade stalling concerns due to high loading. It is also assumed that the rotor of a RVR helicopter will operate in or near to an auto-rotative condition (small amount of rotor power at low tip path plane angle) with auxiliary propulsion of the vehicle. [Ewans, 1973] While in this flight condition the stability axis of the aircraft is moved away from the center of the rotor disk to a location closer to that of a typical fixed wing aircraft. This relieves the lateral and longitudinal cyclic pitch constraints since the rotor disk will seek an equilibrium condition based primarily on collective pitch. The final condition that did not allow the RVR system to trim at high speed was rotor drag. With the RVR system in full operation ($\mu = 2$ and negative pitch on the retreating side of the rotor disk) conventional helicopter rotor drag analysis does not work. Fortunately, having the rotor in a near auto-rotative state allows a simple but critical drag assumption; over the entire rotor, the integrated effect of the tilt of the lift vector at each blade element is sufficient to overcome the integrated effect of the drag at all of the blade elements. [Prouty, 1986] Only a small amount of power is required to overcome the drag and provide rotor stability. These assumptions are highly interdependent. All of the conditions must be satisfied concurrently whenever RVR trim calculations are applied.

Since having the correct advance ratio, μ, is critical to successful RVR operation JANRAD 2P_RVR has a built in calculator which computes the optimal μ for the design cruise airspeed such that the entire retreating blade is in reverse flow. This optimal μ value ensures the reverse flow area around the retreating blade is sufficient to support RVR aerodynamic operation and RVR based cyclic pitch, discussed below.

3. RVR Based Cyclic Pitch

One final element had to be added to the RVR system: RVR based cyclic pitch. The use of HHSC pitch is required for advance ratios greater than 1.0. [Zientek, 2001] Using 1/rev cyclic pitch alone, the rotor cannot achieve the required large negative pitch in the 270 degrees blade sector, yet maintain positive pitch everywhere else. See Figure 29.

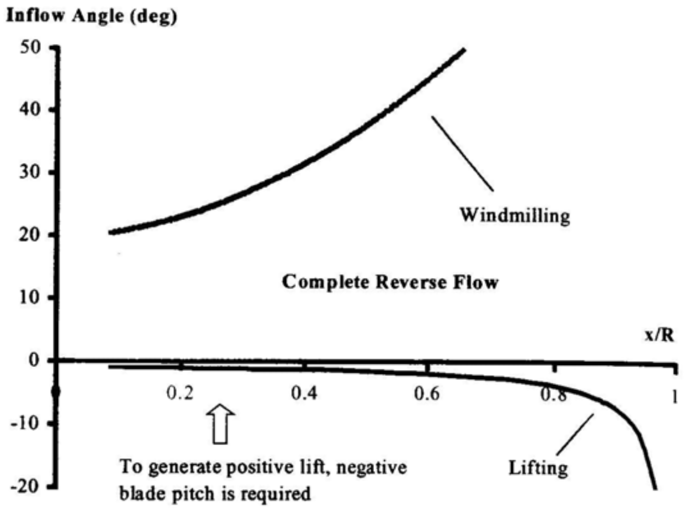


Figure 29. Inflow Angle versus. Blade Station. [From: Zientek, 2001]

The 2/rev RVR based cyclic pitch can obtain the proper blade angle of attack on the retreating side without the aft tilt normally associated with windmilling operation. This differs from conventional HHSC. The equation for θ takes the form

$$\theta_{rvr} = \theta_0 + \theta_0 \sin(\psi) - \theta_{1c} \cos(\psi) - \theta_{1s} \sin(\psi) + \theta_{2c} \cos(2\psi)$$

The negative signs preceding the θ_{1c} and θ_{1s} terms are required to move the maximum cyclical inputs to the retreating sections of the rotor disk. Figure 30 compares conventional HHSC based cyclic pitch and RVR based cyclic pitch.

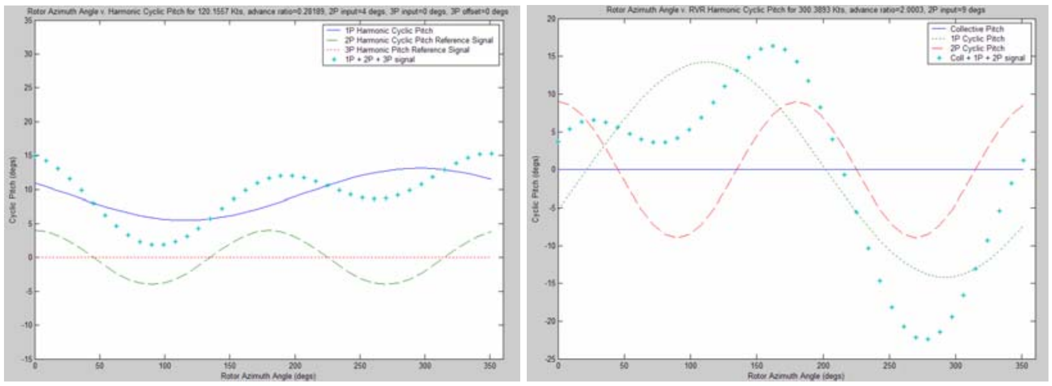


Figure 30. Conventional HHSC Cyclic Pitch (Left) Compared to RVR Based Cyclic Pitch.

D. EXPERIMENTAL RESULTS

All of these assumptions are enforced within JANRAD 2P_RVR by implementing user enabled logic gates. The logic gates allow JANRAD 2P_RVR to operate in either conventional or RVR based cyclic pitch. JANRAD 2P_RVR only contains one non-user driven restriction for RVR based cyclic pitch employment, airspeed must be greater than 200 knots. Figures 31 through 35 show JANRAD 2P_RVR output.

Figure 31 clearly shows the effects of the RVR Based Cyclic Pitch. The negative blade angles at rotor azimuth positions between 230 degrees and 320 degrees are crucial to the development of positive lift. Also note that collective pitch has been reduced to zero through JANRAD 2P_RVR's trim constraints, complementing the assumption that the rotor system has entered a near autorotative state.

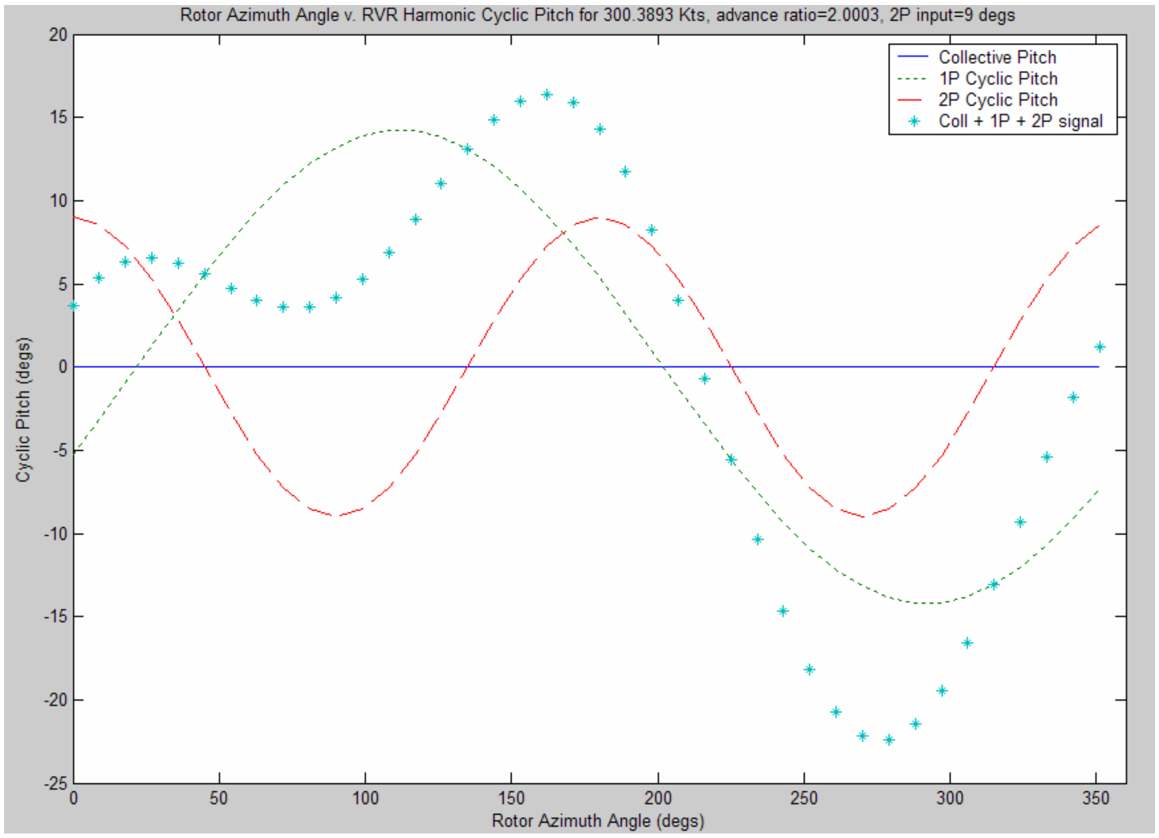


Figure 31. Nature of RVR Based Cyclic Pitch.

Figure 32 shows the Mach number distribution around the rotor disk. Note that the retreating side of the blade, approximate blade azimuth positions of 200 degrees to 340 degrees, is fully entrenched in reverse flow. It is the combination of this reverse flow, the RVR's local angle of attack, and C_L that allow for positive lift distribution.

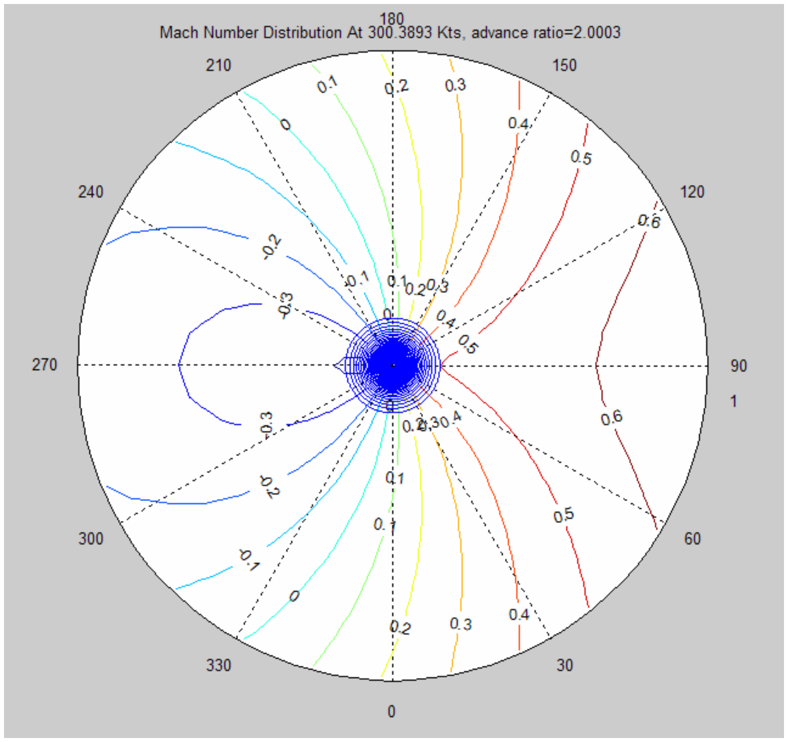


Figure 32. Rotor Disk Mach Number Distribution.

Figure 33 shows C_L distribution viewed as a contour plot. One can immediately see that the disk is trying to achieve equilibrium. HHSC contour lines are replaced with the RVR based cyclic pitch contour lines. The RVR based cyclic pitch builds upon the principles of HHSC but shifts the emphasis on stall control from the forward and aft portions of the disk to the left and right sides of the disk. The shift of stall control to the lateral sections of the disk is critical to take full advantage of the traits of the RVR. The contour lines in Figure 33 clearly illustrate that the RVR based cyclic pitch is functioning as predicted. Also apparent in Figure 33 are the high C_L values in the forward right sector of the disk. This phenomenon is discussed in the following paragraphs. The distinct left and right side loading is further illustrated in Figures 34 and 35.

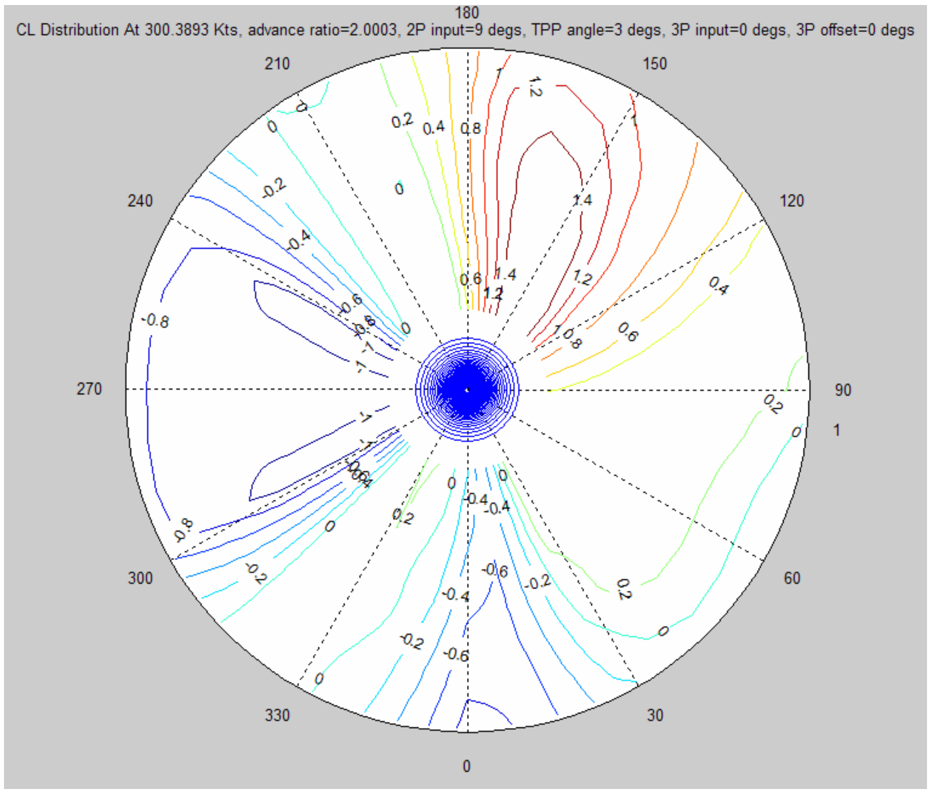


Figure 33. C_L Distribution Contour Plot.

Figure 34 shows the radial distribution of blade lift at the most important rotor azimuth locations. At the $\psi = 270$ degree location positive lift is observed even though the blade is total reverse flow. As stated earlier, the positive air load is a result of the RVR's unique aerodynamic behavior given the proper operating conditions.

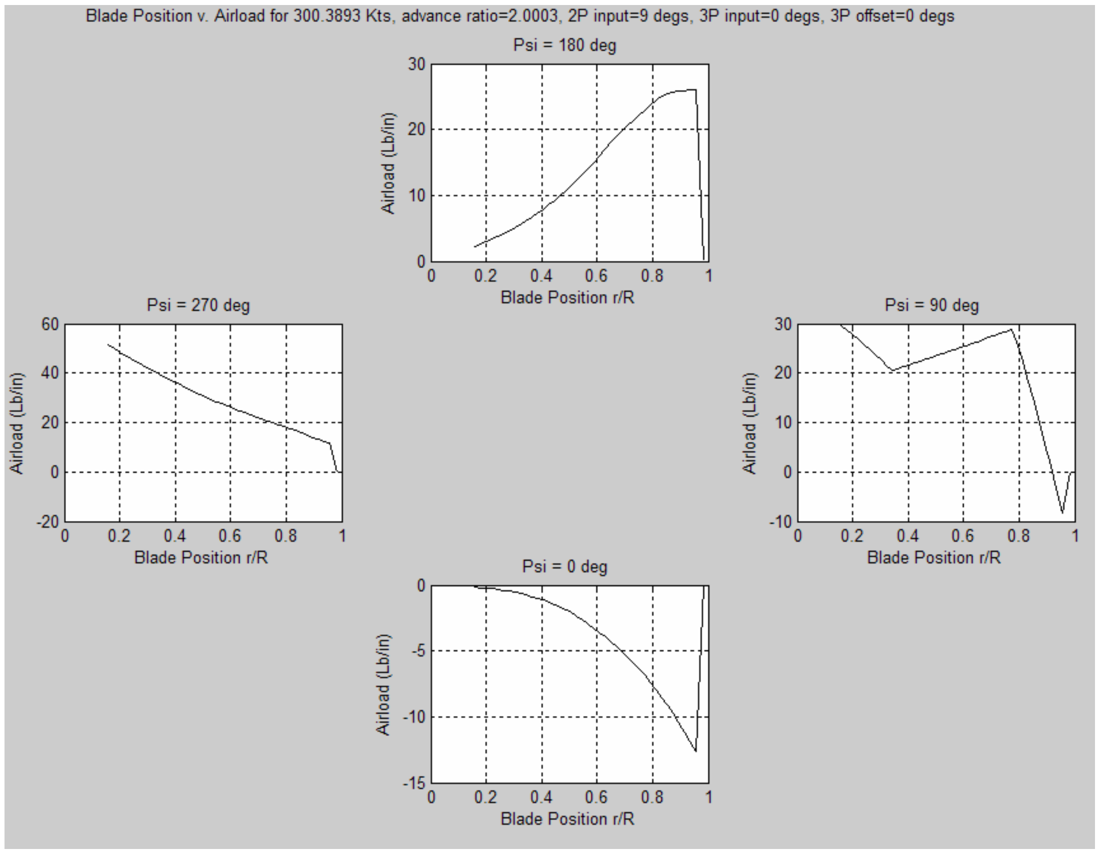


Figure 34. Blade Position versus Air Load at Critical Rotor Azimuth Locations (0, 90,180, and 270 Degrees).

Figure 35 is a three dimensional representation of the lift distribution around the disk. Note the predominant lift 'hump' centered at the disk's 270 degree azimuth location, further proof the RVR based cyclic pitch is performing as predicted. Also note the 'spike' centered at approximately ψ =160 degrees. This 'spike' is the result of high blade angle, and therefore high blade coefficient of lift. The high blade angle is due to the cumulative effects of the cyclic inputs within that azimuth quadrant. Although undesirable, this phenomenon does not effect the disk's ability to trim within the computational boundaries established in JANRAD 2P_RVR.

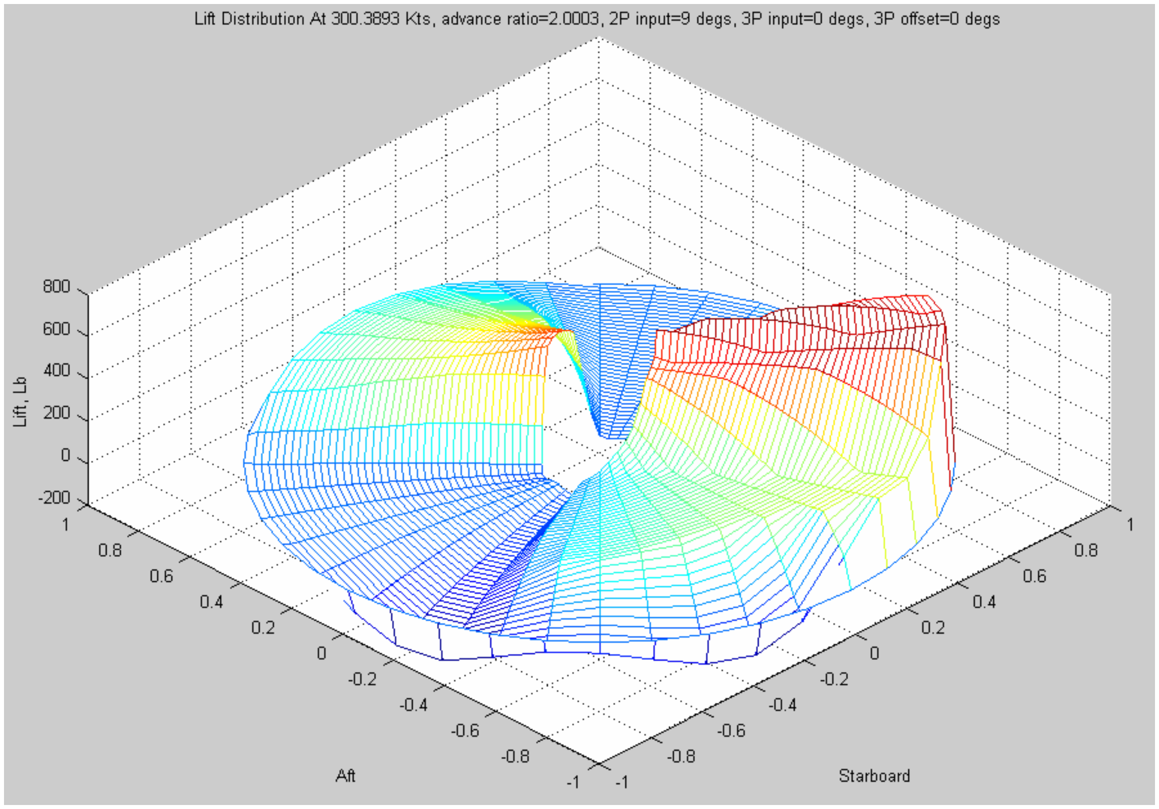


Figure 35. Lift Distribution Surface Plot.

This area of abnormal lift distribution will certainly cause vibrations and high blade stress as each rotor blade is forced to pass through the effected azimuths. In fact, Ewans encountered similar unbalanced lift forces in his tests [Ewans, 1975] that made it impossible to complete his entire test program. In an effort to smooth the lift distribution a 3P signal (4 degrees) was added. The resulting lift distribution is shown in Figure 36. In this figure three distinct humps are visible and the overall lift distribution is generally equalized between the three humps. Application of additional higher harmonic inputs would continue to suppress unwanted lift spikes. Detailed study of rotor system behavior using harmonics higher than 3P is beyond the scope of this study and would require further analysis in future research.

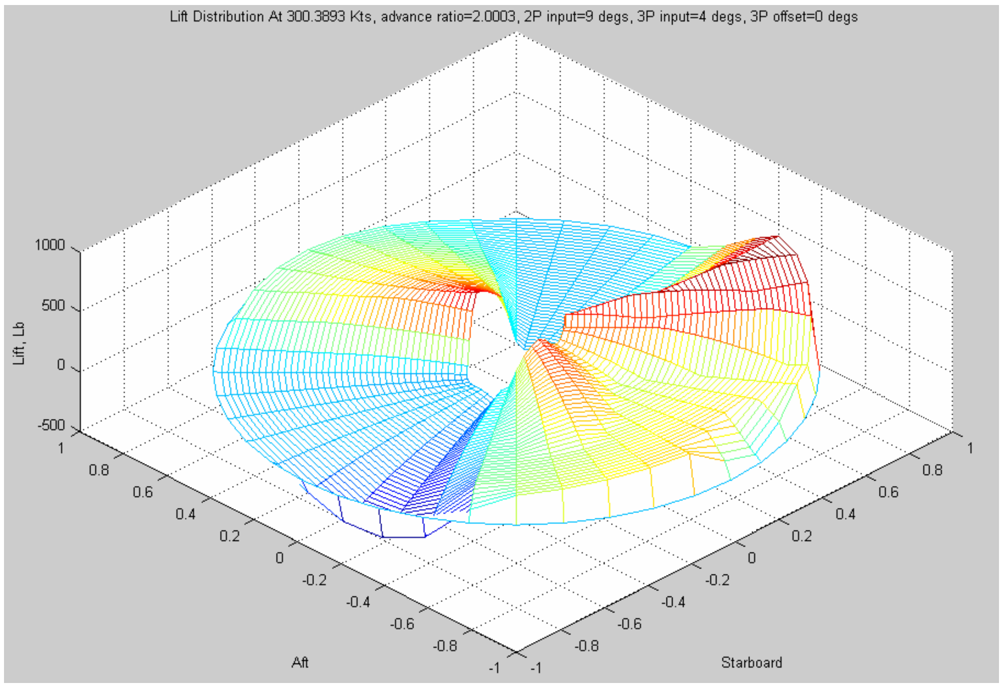


Figure 36. Lift Distribution Surface Plot with 3P Harmonic Cyclic Pitch.

JANRAD 2P_RVR has shown that a RVR system with tailored 2P cyclic pitch does trim under the assumptions used in the study. The ability to trim allows for sustained high speed aerodynamic flight without incurring high drag penalties or autogiro type behaviors. The ability to maintain aerodynamic flight with a rotor system above 200 knots allows designers to expand the performance and usability of current helicopters. The next section explores one possible way to employ a complete RVR system to fulfill an emerging vertical heavy lift requirement.

IV. APPLICATION OF THE RVR AND ACCOMPANYING CONTROL SYSTEM TO THE JOINT VERTICAL HEAVY LIFT (JVHL) AIRCRAFT

A. FIRST ITERATION DESIGNS TO SATISFY REQUIREMENTS

In an effort to satisfy operational requirements defined in the first chapter, two 'new-start' aircraft, the Joint Heavy Lift (JHL) and Reverse Velocity Rotor (RVR), were the subjects of simultaneous preliminary design analysis. For more information on these advanced designs the reader is referred to the references entitled "Joint Heavy Lift Expeditionary Warfare Aircraft Solution" [Wood and Aaron, 2003] and "Reverse Velocity Rotor Approach for Design of a STOVL Heavy Lift Transport for Expeditionary Warfare." [Wood and Van Riper, 2003] The JHL is intended to be a long range, heavy lift aircraft to support Marine Corps, joint, and coalition force operations ashore up to 200 NM inland in a forcible entry environment. The JHL is a Joint Strike Fighter (JSF) lift-fan variant modified for the increased load-carrying requirement. Large lift fans, each producing up to 60,000 pounds of vertical thrust, are embedded in the wing sections of the aircraft to provide Vertical Take-off and Landing (VTOL) capability while thrust vectored turbofan engines provide forward thrust while in forward flight. The RVR aircraft is a compound helicopter equipped with a conventional anti-torque tail rotor and large turboprop engines to provide the auxiliary thrust required for high-speed forward flight. The rotor system is an eight bladed (110 foot diameter), fully articulated, foldable system with each blade incorporating the RVR airfoil cross section. The rotor system is driven by a lightweight variable speed transmission that allows RPM to be set at either 100% or 50% of normal operating RPM based on flight mode. The anti-torque system is a traditional six bladed pusher tail rotor mounted to the tail vertical pylon. Auxiliary propulsion is provided by two propellers (fifteen foot diameter), each driven by one of the two wing-mounted turboprop engines. Figures 37 and 38 show each design.

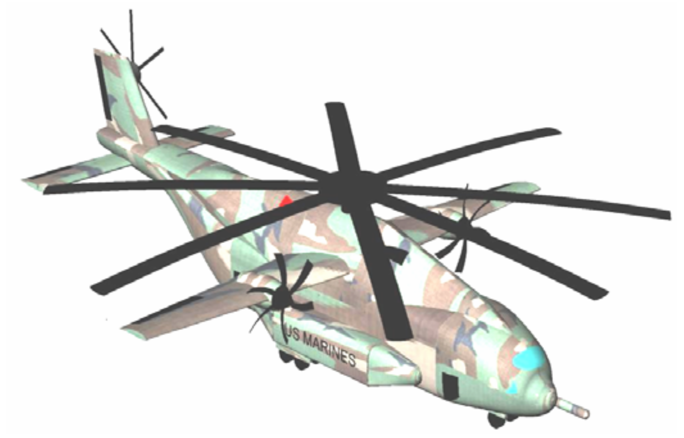


Figure 37. RVR Based Aircraft.

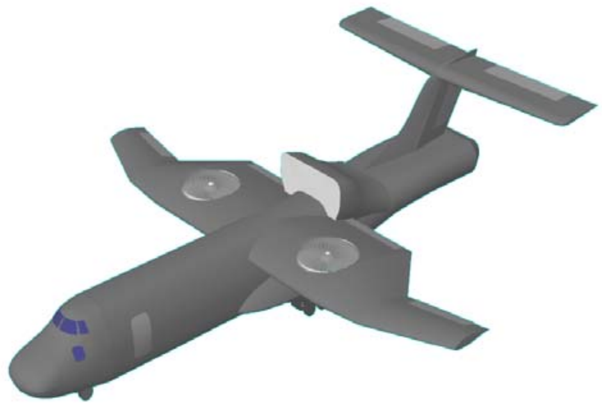


Figure 38. JHL Aircraft.

Both aircraft are intended to operate from ships of the amphibious task force, specifically the Amphibious Assault and Logistics Support variants of the family of ExWar platforms currently under design by the Total Ship Systems Engineering (TSSE) curriculum at the Naval Postgraduate School. The aircraft are also designed to be compatible with legacy force ships, i.e. LHD, LHA (R), and MPF (F) class ships to the maximum extent possible.

As might be expected these evolutionary designs became the subject of much debate and criticism. Concerns about the RVR ranged from the expected size, weight, and complexity of a variable speed main transmission able to handle an aircraft with a take-off gross weight of over 100,000 pounds to the employment of a tail rotor almost as large as a UH-60 main rotor blade. The JHL suffered from lift fan and wing integration challenges and a downwash velocity, the vertical wind produced as the aircraft hovers, which could practically lift concrete blocks off the ground and send them flying several yards away. Clearly, a more amiable solution had to exist.

B. THE JOINT VERTICAL HEAVY LIFT AIRCRAFT

Ongoing research has produced a new conceptual design that combines the best traits of the RVR and JHL aircraft while greatly reducing the negative aspects of each design. Although the JHL did not employ any RVR or HHSC technologies that form the core of this study, it provides the framework for the F119 (JSF) Shaft Driven Lift Fan and Propulsion module integration. This technology provides the key to overcoming the most challenging design point of the aircraft; sustained hover and low speed flight in harsh environments. The proposed hybrid design, labeled the Joint Vertical Heavy Lift (JVHL) uses a RVR based rotor system and drive train with three F-35 V/STOL type Shaft Driven Lift Fan and Propulsion Modules. A direct jet anti-torque thruster is used in lieu of a conventional anti-torque tail rotor. JANRAD 2P_RVR was used exclusively for the JVHL's rotor performance calculations. A detailed description of JANRAD 2P_RVR and user instructions are provided in Chapter V. The proposed configuration of the JVHL is shown in Figures 39-42. Dimensions shown in Figure 39 are in feet.

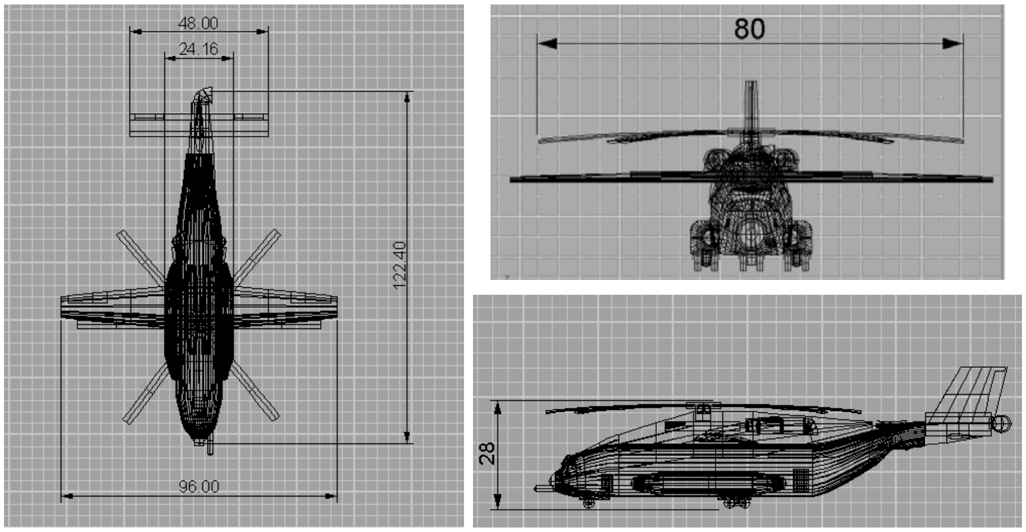


Figure 39. JVHL Three View.

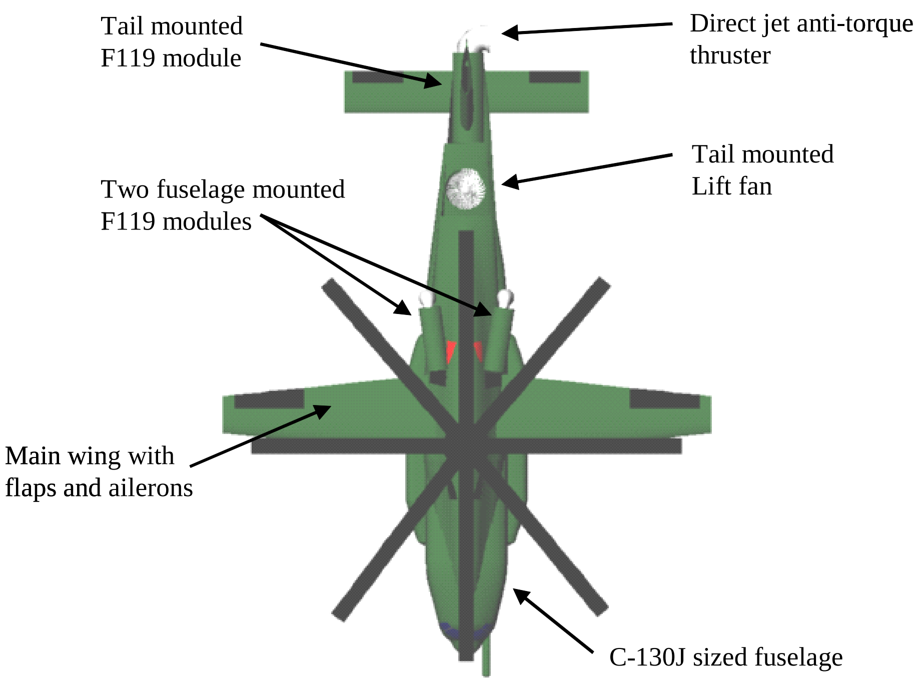


Figure 40. Top View of JVHL.

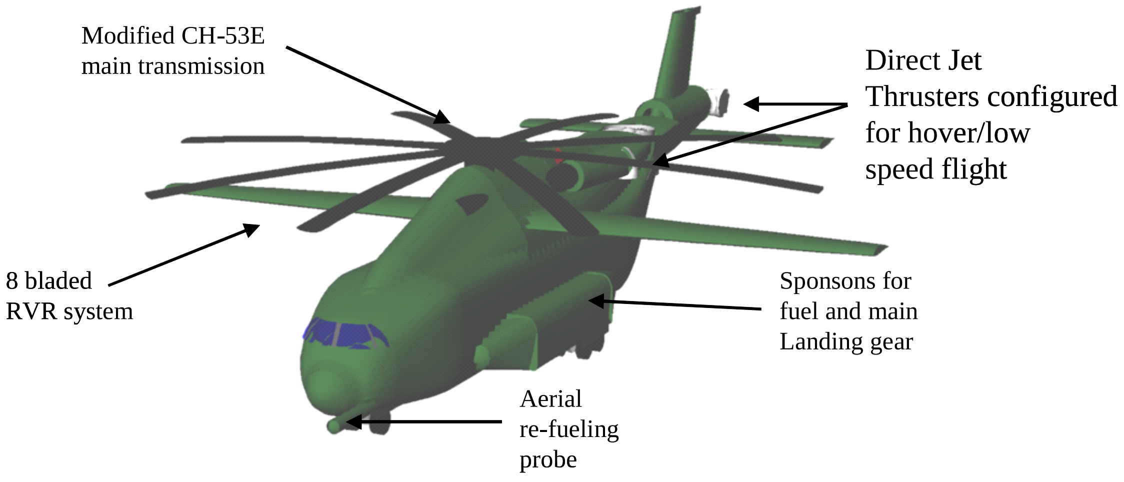


Figure 41. Perspective View of JVHL.

Figure 42 shows the JVHL in hover configuration with 60,000 lbs of lift from the main rotor, 18,000 pounds from each fuselage mounted F119 module and 18,000 pounds from the tail mounted shaft driven lift fan, for a total of 114,000 lbs of vertical lift.

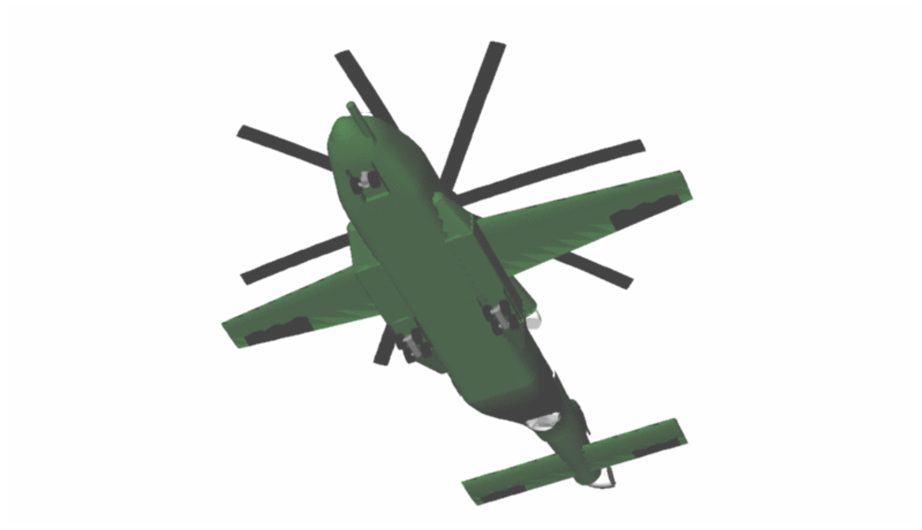


Figure 42. JVHL in Hover Configuration.

Immediate benefits of using the F119 propulsion modules are clearly evident. Most important, the propulsion module offsets the hover and low speed performance of the RVR system by providing a great deal of vertical thrust. Current F119 engine configurations would provide ample power for all flight modes of the JVHL. In fact, the three F119 modules would provide such a surplus of power that each module could be down-rated by up to 30%. This modification would undoubtedly reduce fuel requirements and, more importantly, reduce engine fatigue.

1. Fuselage Modules

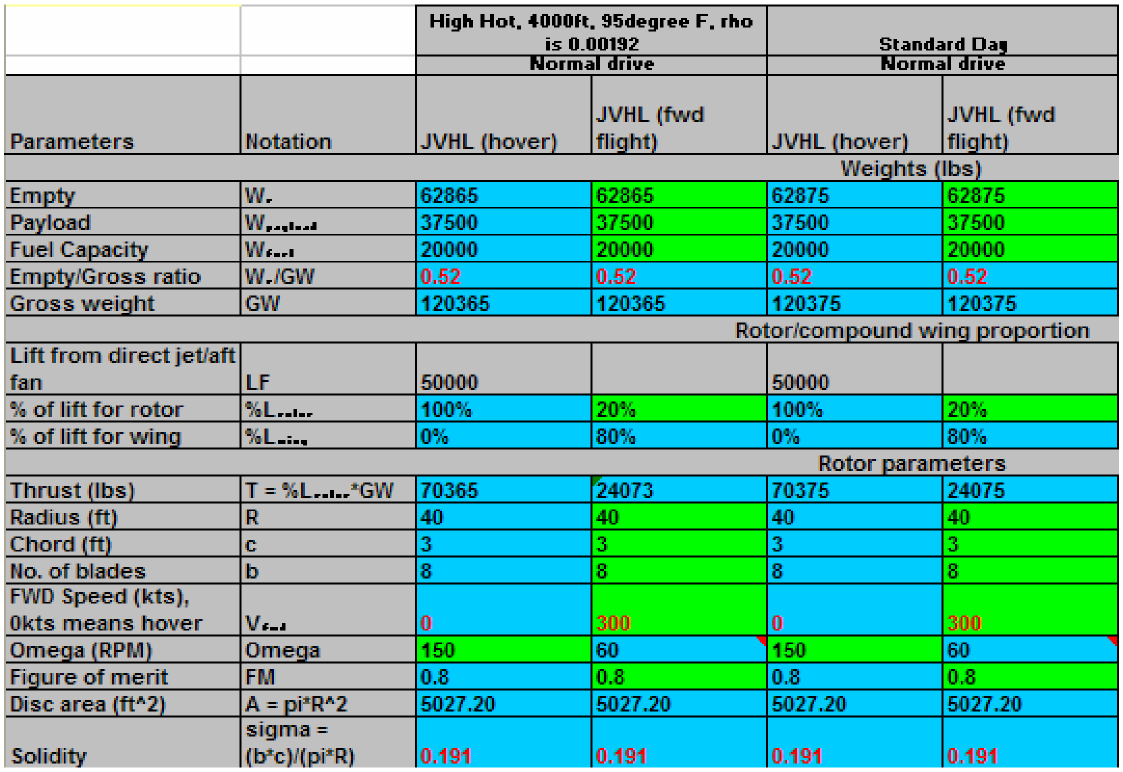
The fuselage mounted modules are installed so that the drive shafts leading to the main transmission have a straight path and direct jet exhausts do not have to be ducted around the fuselage. When in forward flight the two fuselage mounted modules provide ample thrust so the JVHL can achieve the design speed of 300 knots while at an altitude of 10,000 feet MSL. The large compound wing no longer has to accommodate turboprop engines or complicated shafting. The wing can now have a full complement of control surfaces and even be modified to carry extra fuel. Of course the wing will still have a negative effect on hover performance but the added vertical thrust provided by the direct jet thrusters and the aft lift fan help reduce the wing's adverse effects.

2. Tail Module

The tail module eliminates all tail rotor components and replaces them with a self-contained highly reliable vertical lift augmentation and anti-torque solution while significantly reducing drag in high-speed flight. The vectored thrust nozzle is collective and pedal coupled to ensure maximum anti-torque thrust and directional control is delivered proportional to inputs. While hovering, the module would also provide approximately 20,000 lbs vertical lift via the built in Lift Fan. This lift would help offset main rotor lift and power requirements and extend the CG range of the aircraft.

3. Preliminary Design Data

Another benefit of the main rotor, direct jet thruster, and left fan configuration is that the downwash will be less than half that of the JHL and the direct jet thruster output will be almost immediately cooled by the entrained flow circulating through the main rotor system. One of the most important advantages of using the F119 module is that now the JVHL will have a measurable power and performance margin that will enable the aircraft to hover with a full load on a High/Hot day (4000 ft MSL and 95 degrees F). Critical design data are shown in Tables 1 and 2.



		High Hot, 4000ft, 95degree F, rho is 0.00192		Standard Day	
		Normal drive		Normal drive	
Parameters	Notation	JVHL (hover)	JVHL (fwd flight)	JVHL (hover)	JVHL (fwd flight)
			Weights (lbs)		
Empty	W_e	62865	62865	62875	62875
Payload	$W_{payload}$	37500	37500	37500	37500
Fuel Capacity	W_{fuel}	20000	20000	20000	20000
Empty/Gross ratio	W_e/GW	0.52	0.52	0.52	0.52
Gross weight	GW	120365	120365	120375	120375
			Rotor/compound wing proportion		
Lift from direct jet/aft fan	LF	50000		50000	
% of lift for rotor	$\%L_{rotor}$	100%	20%	100%	20%
% of lift for wing	$\%L_{wing}$	0%	80%	0%	80%
			Rotor parameters		
Thrust (lbs)	T = $\%L_{rotor}$*GW	70365	24073	70375	24075
Radius (ft)	R	40	40	40	40
Chord (ft)	c	3	3	3	3
No. of blades	b	8	8	8	8
FWD Speed (kts), 0kts means hover	V_{fwd}	0	300	0	300
Omega (RPM)	Omega	150	60	150	60
Figure of merit	FM	0.8	0.8	0.8	0.8
Disc area (ft^2)	A = pi*R^2	5027.20	5027.20	5027.20	5027.20
Solidity	sigma = (b*c)/(pi*R)	0.191	0.191	0.191	0.191

Table 1. Basic JVHL Design Data.

		High Hot, 4000ft, 95degree F, rho is 0.00192		Standard Day	
		Normal drive		Normal drive	
Parameters	Notation	JVHL (hover)	JVHL (fwd flight)	JVHL (hover)	JVHL (fwd flight)
Tip speed at 0 deg rotor azimuth	$V_{tip,0}$ = Omega*R	628.40	251.36	628.40	251.36
Tip speed at 90 deg rotor azimuth	$V_{tip,90}$ = Omega*R + V_{fwd}	628.40	758.36	628.40	758.36
Tip speed at 180 deg rotor azimuth	$V_{tip,180}$ = Omega*R	628.40	251.36	628.40	251.36
Tip speed at 270 deg rotor azimuth	$V_{tip,270}$ = Omega*R - V_{fwd}	628.40	-255.64	628.40	-255.64
Max Mach$_{tip}$	Max M_{tip}	0.57	0.68	0.57	0.68
Disc loading (psf)	DL = T/A	14.00	4.79	14.00	4.79
Induced velocity (fps)	V_i = sqrt(DL/2*rho)	60.37	Not applicable	54.25	Not applicable
Advance ratio	mu = V_{fwd}/Omega*R	0.00	2.01	0.00	2.01
$P_{induced}$(HP)	$P_{induced}$(HP) = T*V_i / 550	7724.02	Not applicable	6941.94	Not applicable
1st approx: P_{actual}(HP)	P_{actual}(HP) = $P_{induced}$ / FM	9655.03	Not applicable	8677.43	Not applicable
Thrust coeff. / sigma	C_T = T/A*rho*(omega*R)^2*sigma	0.088	Not applicable	0.078	Not applicable
Torque coeff. / sigma	C_Q / sigma (from Proutly text page 90-92)	0.0270	Not applicable	0.020	Not applicable
Torque	Q =	786083	Not applicable	721183	Not applicable
2nd approx: P_{actual}(HP)	calculated from P = Q*omega/550 where Q is from Cq/sigma curve in text page 90-92	22450	Not applicable	20597	Not applicable
plus 33%	Final P required	29859	Not applicable	27394	Not applicable

Table 2. JVHL Rotor and Power Calculations.

C. RVR ANALYSIS

The JVHL uses two emerging technologies, the RVR system and the F119 propulsion module. All other major components of the aircraft are based on fairly mature technologies and processes. Compound helicopter design is well understood, the CH-53E rated transmission and drive line technology needed to satisfy main rotor lift requirements has been refined by the helicopter industry, and use of composites in fuselage design practices is becoming more commonplace with each passing year. The only remaining design obstacle is the RVR rotor system.

A detailed aeroelastic and dynamic analysis of the RVR system is beyond the scope of this study but a preliminary aerodynamic analysis was conducted to ensure the RVR system would be viable at high airspeeds. JANRAD version 1P2P3Prvr was used to check rotor system performance. As discussed earlier, correctly manipulating the advance ratio is critical since the RVR system relies on the entire retreating side of the rotor system being fully entrenched in reverse airflow. For the JVHL an advance ratio of 2 is used at the aircraft's cruise speed of 300 knots. To achieve this advance ratio the rotational speed of the rotor system is reduced from 150 rpm to 60 rpm. This high advance ratio causes sufficient reverse airflow on the retreating side of the rotor disk, trailing edge to leading edge, to produce lift. The RVR is coupled with a RVR unique two per revolution (2/rev) cyclic pitch signal. See Figures 43 – 45.

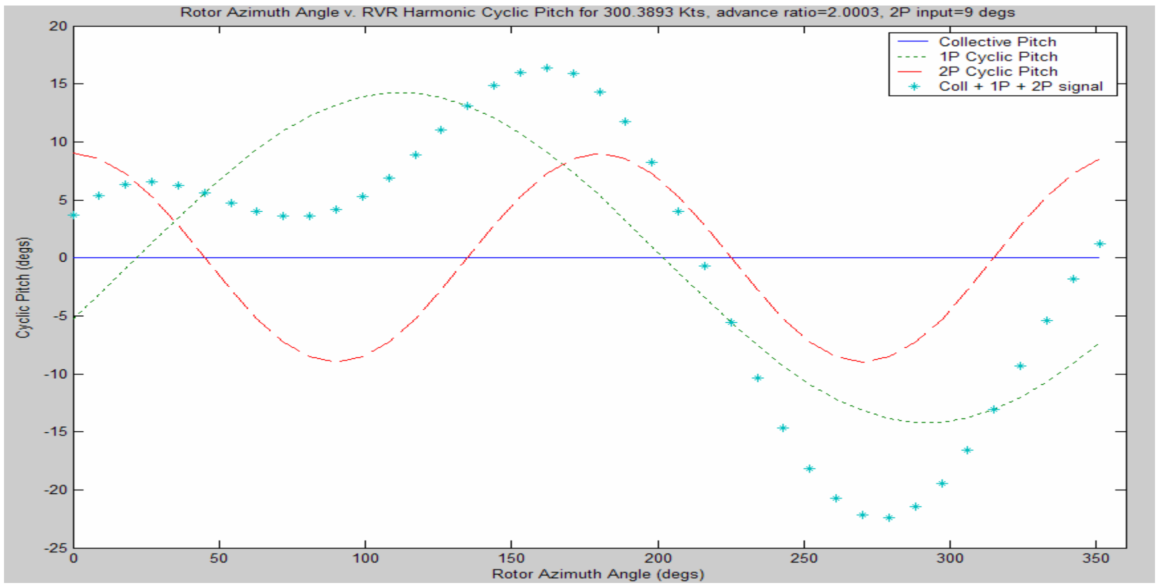


Figure 43. RVR Based Cyclic Pitch for the JVHL.

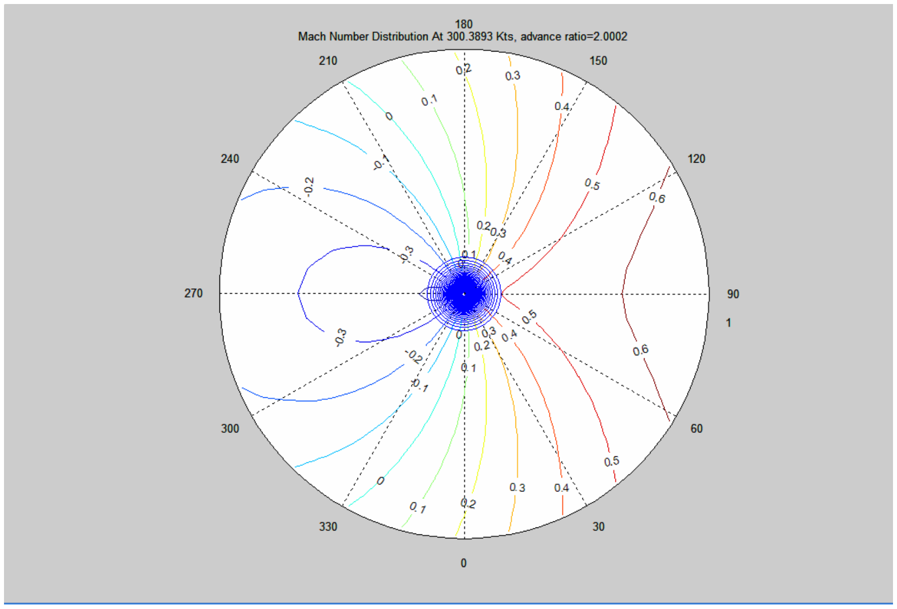


Figure 44. Mach Number Distribution at Cruise Speed (300 kts),Retreating Side of Blade is in Reverse Flow.

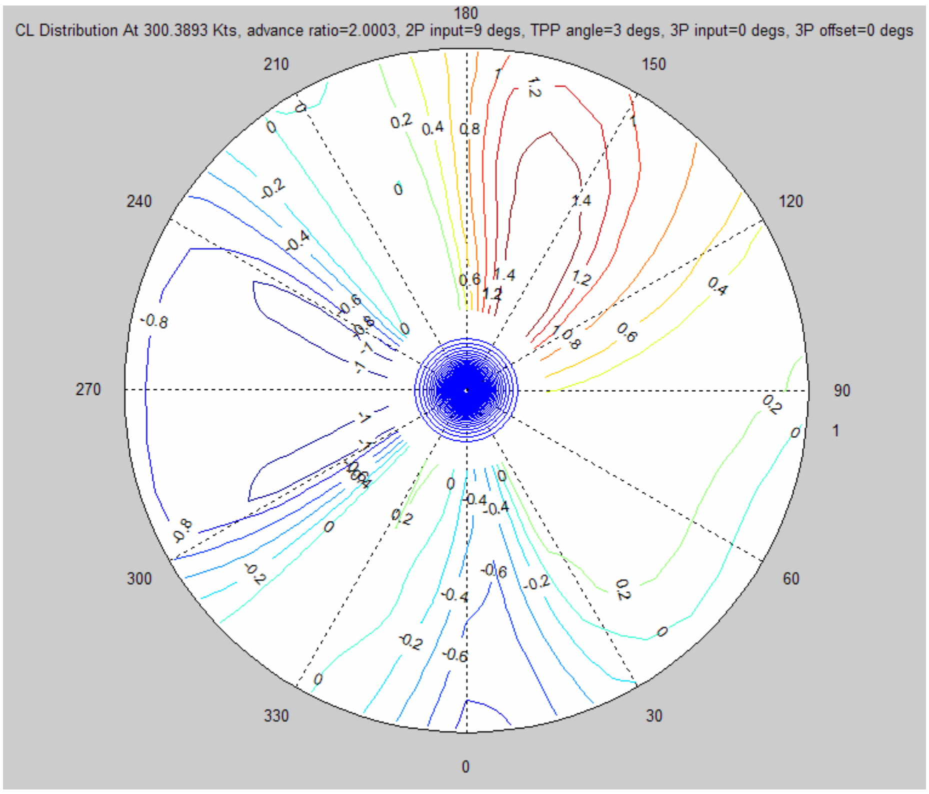


Figure 45. C_L Distribution at Cruise Speed (300 knots) for the JVHL.

Since the large compound wing has unloaded the rotor (at cruise speed the wing is carrying 80% of aircraft weight and the rotor is carrying the remaining 20%), the rotor disk can operate in an autorotative state with small cyclic input and no wind milling. Without this wind milling behavior there is almost no drag penalty. These figures illustrate the behavior of this rotor system is directly in agreement with predicted RVR behavior. The aircraft is able to operate at high cruise speed without the typical negative effects of the rotor disk.

V. JANRAD USER INSTRUCTIONS

A. GENERAL

Program installation, guidance for input requirements, and rotor performance output are discussed in the subsequent sections. These instructions assume that the reader has a working knowledge of MATLAB release 12 and higher.

Joint Army Navy Rotorcraft Analysis and Design (JANRAD) version 1P2P3Prvr was developed to allow introduction of 2/rev cyclic inputs, 3/rev cyclic inputs, compound wing sizing, unconventional vertical lift augmentation, and employment of the Reverse Velocity Rotor (RVR) system and subsequent RVR specific 2P control inputs. This version also includes a comprehensive plotting capability that gives the user high fidelity 2-D, 3-D, and contour plots to aid in rotor performance analysis.

JANRAD version 1P2P3Prvr was built using the original source code found in JANRAD version 1 [Nicholson, 1993]. This approach ensured the original trim routine that forms the heart of program was unaltered by subsequent undocumented updates.

The program was developed and tested on a Pentium 4 based laptop running MATLAB release 13 under Windows XP Professional. The laptop was equipped with 512 megabytes of RAM and a twenty gigabit fixed drive.

B. INSTALLATION

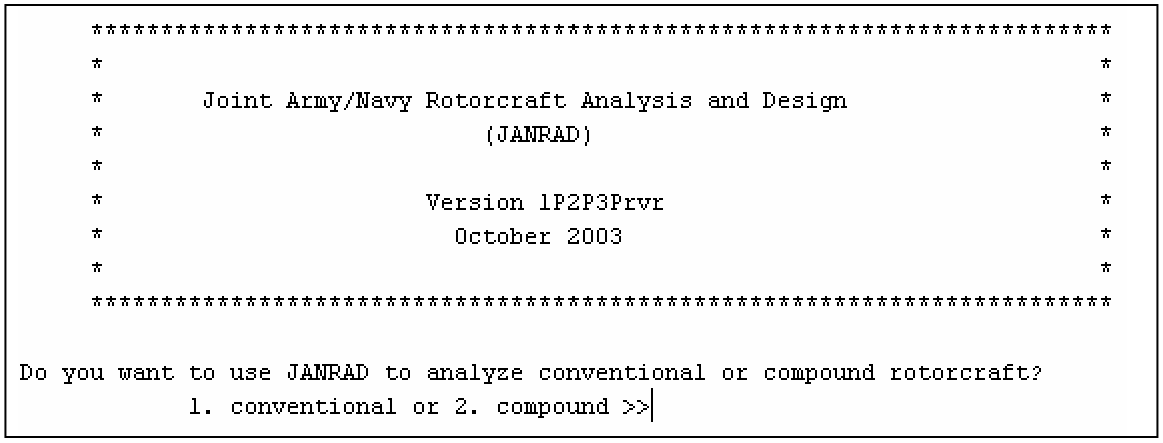
To install JANRAD 2P_RVR, create a subdirectory named JANRAD and install the following JANRAD 2P_RVR modules: janrad.m, trim.m, dmcalc.m, hh02clcd.m, oo12clcd.m, perf.m, plotter.m, polar1.m, rvrclcd.m, sc1095r8clcd.m, thrcalc.m, tmcalc.m, and vr12clcd.m. Listings of the modules can be found in subsequent appendices.

Once this installation is complete, open MATLAB and type JANRAD (with the proper drive designation) in the Current Directory dialogue box. All of the modules must be installed in the select directory on the computer's fixed drive for the program to function. The program will not work if the user attempts to run the modules from a remote drive, compact disk, or network. The program is now ready to run.

C. PROGRAM EXECUTION

1. Input

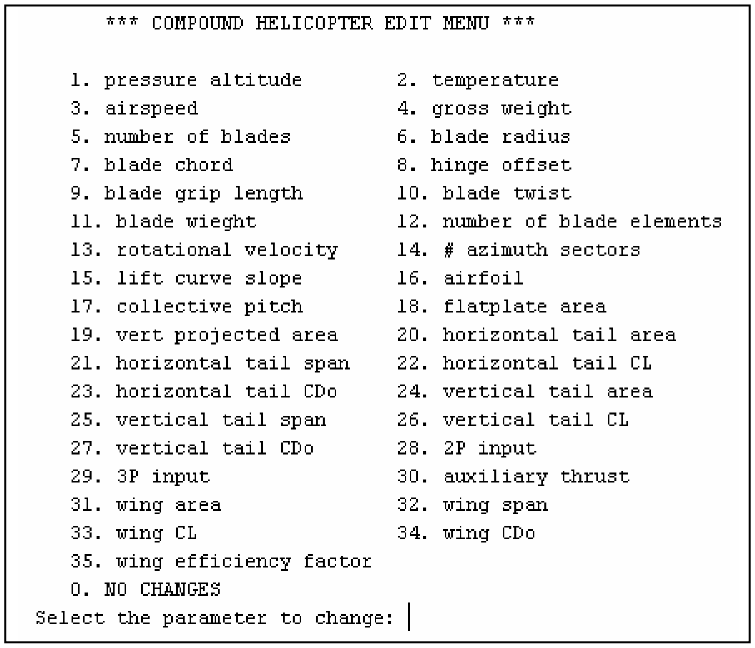
To run JANRAD 2P_RVR beta, at the MATLAB command prompt, type 'janrad'. At the Title page enter a 1 if you want to perform analysis on a conventional helicopter or 2 if you want to perform analysis on a compound helicopter. RVR based analysis is only allowed if you select the compound helicopter option. See Figure 46.



```
     ************************************************************************
     *                                                                      *
     *       Joint Army/Navy Rotorcraft Analysis and Design                 *
     *                            (JANRAD)                                  *
     *                                                                      *
     *                         Version 1P2P3Prvr                            *
     *                           October 2003                               *
     *                                                                      *
     ************************************************************************

Do you want to use JANRAD to analyze conventional or compound rotorcraft?
            1. conventional or 2. compound >>
```

Figure 46. Title Page.

At the File Selection page, enter a 1 if you want to edit a previously stored data file, otherwise enter a 2. If you chose to edit a file, you will be prompted on the next page to enter the file name without the extension. If JANRAD 2P_RVR is unable to find the specified file you will be prompted to try again. Ensure your spelling is correct and you are in the proper directory. Once the file is loaded the edit menu, as seen in Figure 47, will be displayed. Check to make sure the menu's title corresponds to the type of helicopter you are analyzing, either conventional or compound.



```
        *** COMPOUND HELICOPTER EDIT MENU ***

     1. pressure altitude        2. temperature
     3. airspeed                 4. gross weight
     5. number of blades         6. blade radius
     7. blade chord              8. hinge offset
     9. blade grip length        10. blade twist
     11. blade wieght            12. number of blade elements
     13. rotational velocity     14. # azimuth sectors
     15. lift curve slope        16. airfoil
     17. collective pitch        18. flatplate area
     19. vert projected area     20. horizontal tail area
     21. horizontal tail span    22. horizontal tail CL
     23. horizontal tail CDo     24. vertical tail area
     25. vertical tail span      26. vertical tail CL
     27. vertical tail CDo       28. 2P input
     29. 3P input                30. auxiliary thrust
     31. wing area               32. wing span
     33. wing CL                 34. wing CDo
     35. wing efficiency factor
     0. NO CHANGES
  Select the parameter to change: |
```

Figure 47. Compound Helicopter Edit Menu Page.

Select the number of the parameter you wish to change. The current value for that parameter will be displayed followed by a prompt to enter the new value. If you chose not to change the value, press <ENTER> and the previous value will be maintained. Once a new value is entered or <ENTER> is pressed the edit menu will be displayed again. Choose the number of the next parameter to change or 0 to exit the programs direct edit mode. If you chose to create a new data file, prompts for individual parameters will be displayed. Enter the value of each parameter when prompted.

Enter the values for the specified parameters regardless of editing or creating a data file in the following manner:

1. PA – Enter the pressure altitude in feet
2. temperature – Enter the temperature in degrees Fahrenheit
3. airspeed – Enter forward airspeed in knots. JANRAD automatically converts airspeed to feet/second.
4. GW – Enter aircraft gross weight in pounds.
5. number of blades – Enter the number of blades used in the rotor system.

6. radius – Enter the rotor blade radius in feet. Measure the radius from the center of the rotor hub to the tip of the bade.

7. chord – Enter the average rotor blade chord in feet.

8. hinge offset – Enter the effective hinge offset in feet.

9. blade grip – Enter the distance from the center of the rotor hub to the beginning of the aerodynamic portion of the rotor blade in feet.

10. blade twist – Enter rotor blade twist in degrees. You can enter blade twist as either a positive or negative value. JANRAD uses the absolute value of twist.

11. blade weight – Enter the weight of a single rotor blade in pounds.

12. rotor blade elements – Enter the number of rotor blade radial elements. Multiples of 20 are recommended, i.e. 20, 40, 80. As the number of blade elements increases program execution is slowed. JANRAD adds one additional element to account for tip loss.

13. rotational velocity – Enter the rotor rotational velocity in radians/second.

14. azimuth sectors – Enter the number of rotor disk azimuth sectors. This number must match the number of rotor blade elements if JANRAD created plots are to be used. The number may be different if no plots are required.

15. lift curve slope – Enter the average slope of the linear portion of the rotor blade's airfoil's lift curve, where the lift curve is C_L versus alpha.

16. rotor blade airfoil – Four airfoils are given for use with conventional helicopters and five airfoils are given for use with compound helicopters. Enter the number that corresponds to your desired airfoil.

17. collective pitch – Enter the estimated collective pitch at 0.7 r/R in degrees. JANRAD automatically converts the vale to radians.

18. flat plate area – Enter the aircraft equivalent flat plate area in square feet.

19. vertical projected area – Enter the vertical projected area of the aircraft in square feet.

20. horizontal tail area – Enter the horizontal tail area in square feet. If no horizontal tail, enter a 0.

21. horizontal tail span – Enter the horizontal tail span in feet.

22. horizontal tail C_L – Enter the expected lift coefficient for the horizontal tail.

23. horizontal tail C_{Do} – Enter the horizontal tail profile drag coefficient.

24. vertical tail area – Enter the area of the vertical tail in square feet. If no vertical tail, enter a 0.

25. vertical tail span – Enter the span of the vertical tail in feet.

26. vertical tail C_L – Enter the expected lift coefficient for the vertical tail.

27. vertical tail C_{Do} – Enter the vertical tail profile drag coefficient.

28. 2P longitudinal input – Enter the 2/revolution fixed cyclic input.

29. 3P input – Enter the 3/revolution fixed cyclic input. Enter 0 for no 3P input. Regardless of 3P input you must enter an offset value between 0 and 119 degrees.

30. auxiliary thrust – Enter the expected auxiliary thrust in pounds.

31. wing area – Compound helicopter edit menu only. Enter the wing area in square feet.

32. wing span – Compound helicopter edit menu only. Enter the wing span in feet. If you included the segment of fuselage encompassed by the wing in the wing area value include it in the span value also.

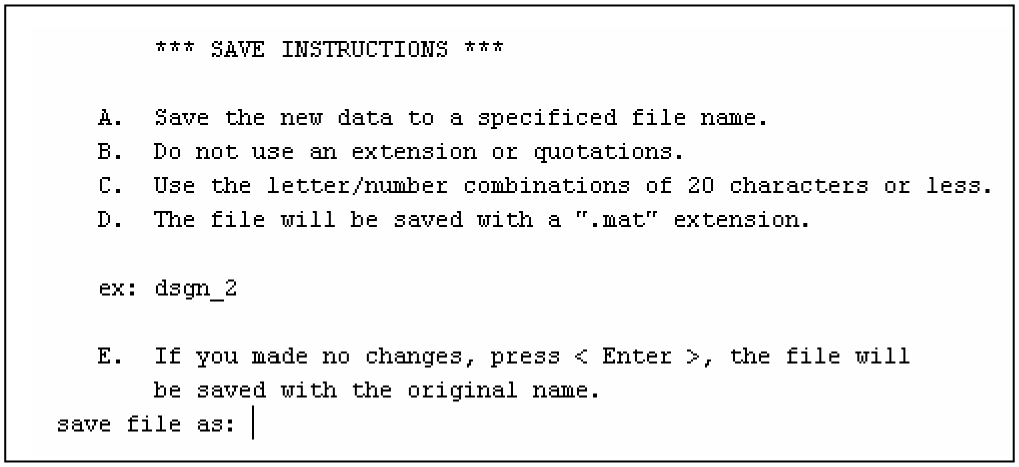
33. wing C_L – Compound helicopter edit menu only. Enter the expected lift coefficient for the wing.

34. wing C_{Do} – Compound helicopter edit menu only. Enter the wing profile drag coefficient.

35. wing efficiency factor – Compound helicopter edit menu only. Enter the wing's expected efficiency factor.

If you have chosen Compound Helicopter Analysis you will be prompted to answer three additional questions: 1. Do you want to fix the Tip Path Plane? 2. Do you want JANRAD to set thrust equal to total drag? 3. Do you want JANRAD 2P_RVRto use RVR based 2/rev Cyclic Pitch? It is recommended that the tip path plane be fixed for compound helicopters. If you chose to fix the angle you will be prompted to enter the angle in degrees. If you chose not fix the angle, the program will modulate the angle to achieve desired performance characteristics. If you chose not to set auxiliary thrust to total drag JANRAD use whatever value was entered in the edit menu. Only chose to use 2/rev cyclic pitch if you are performing high speed (greater than 190 kts) slowed rotor analysis. If Conventional Helicopter Analysis was selected, these questions are bypassed and you are taken directly to the save instructions page.

After all the required parameters are entered, you will be prompted to save the data under a user designated file name. The file name may be up to twenty characters long. JANRAD 2P_RVR will save the file to the current directory as the file name with a ".mat" extension. If you were editing a data file, press <ENTER> if you want to save the file with the original file name. Figure 48 shows the Save Instructions page.



```
          *** SAVE INSTRUCTIONS ***

     A.  Save the new data to a specificed file name.
     B.  Do not use an extension or quotations.
     C.  Use the letter/number combinations of 20 characters or less.
     D.  The file will be saved with a ".mat" extension.

     ex: dsgn_2

     E.  If you made no changes, press < Enter >, the file will
         be saved with the original name.
  save file as:
```

Figure 48. Save Instructions Page.

The Execution Menu page is displayed next. Enter 1 if you want to proceed to next step in Rotor Performance Analysis, a 2 if you would like to go back and edit your data further or if you want to create a new file. To quit the program, select 3.

After selecting Rotor Performance Analysis, JANRAD 2P_RVR will ask you several more questions based on user defined aircraft design inputs. Each question is discussed below.

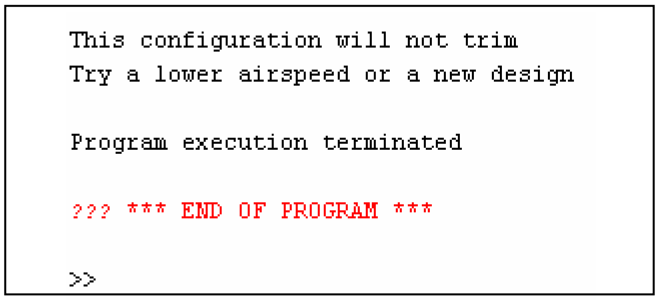
If a Compound Helicopter Analysis was selected, JANRAD 2P_RVR will display the percent of aircraft weight carried by the wing. You may choose to alter the percentage of weight carried by the wing or leave it at the original value. If you decide to change the value the program will calculate and display the new required wing area.

Regardless of Conventional or Compound analysis JANRAD will check to determine if hover/low speed (airspeed less than 10 kts) performance analysis is about to be performed. If hover/low speed analysis has been selected you will be asked if there are any auxiliary vertical thrusters, i.e. direct jet thrusters or lift fans, installed on your aircraft. If installed, enter the amount of thrust in pounds provided by these devices.

If a Compound Helicopter Analysis was selected and you chose not to set auxiliary thrust to total drag, JANRAD 2P_RVR will check to determine if a thrust deficit for you desired flight condition exist. If a deficit exists, it will be displayed and you will be given another chance to set auxiliary thrust equal to total drag. If you decide to continue with the original auxiliary thrust value, a warning message is displayed and the program proceeds to the next analytical step.

If you previously chose to employ 2/rev cyclic pitch, JANRAD 2P_RVR will display the current value of the advance ratio, μ, and if required, suggest altering μ to the optimum value for RVR operation. Advance ratio must be greater than or equal to 1.0 to enable RVR based cyclic pitch. If you allow the program to change μ the rotational velocity of your rotor will be changed to reflect the optimized μ.

After all these inputs have been satisfied, the program will display an ‘ANALYSIS IN PROGRESS’ message and several status messages. If your inputs fail to produced a converged solution using either conventional or RVR trim criteria the message shown in Figure 49 will be displayed.



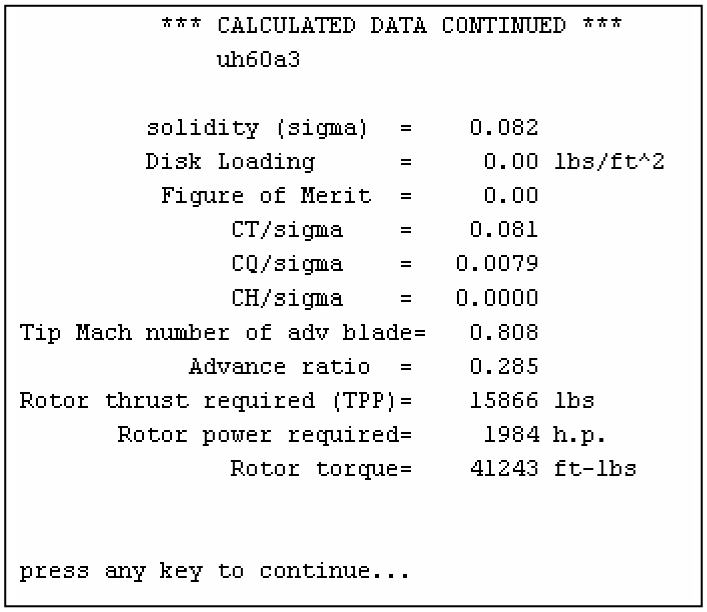
```
This configuration will not trim
Try a lower airspeed or a new design

Program execution terminated

??? *** END OF PROGRAM ***

>>
```

Figure 49. Program Termination Page.

From this page you must restart JANRAD 2P_RVR from the command prompt. A ‘ROTOR TRIMMED’ message will be displayed when the analysis is complete.

2. Output

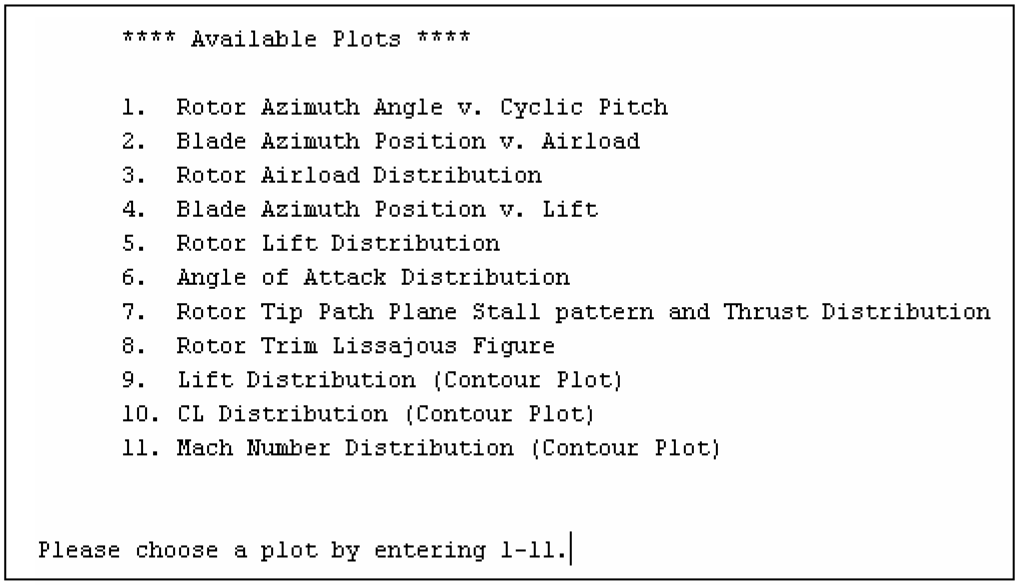
Upon completion of the analysis, JANRAD 2P_RVR will ask if you want the performance results displayed on screen. If you answer ‘yes’ two screens displaying input data and two screens displaying output data will be displayed. The final output screen is displayed in Figure 50.



```
           *** CALCULATED DATA CONTINUED ***
               uh60a3

          solidity (sigma)  =    0.082
          Disk Loading      =     0.00 lbs/ft^2
           Figure of Merit  =     0.00
               CT/sigma     =    0.081
               CQ/sigma     =   0.0079
               CH/sigma     =   0.0000
Tip Mach number of adv blade=    0.808
             Advance ratio  =    0.285
Rotor thrust required (TPP)=    15866 lbs
       Rotor power required=     1984 h.p.
               Rotor torque=    41243 ft-lbs


press any key to continue...
```

Figure 50. Fourth Output Page.

Each of these screens is displayed one at a time. As soon as you advance to the next screen, you cannot return to the previous screen. A copy of the four screens is saved to the current directory with the file name you entered at the Save Instructions page and concatenated with a '.prf' extension. If you answer 'no' the four screens will still be saved and you will be asked if you would like to view any of the JANRAD 2P_RVR created plots. If you answer 'yes' the Available Plots page, seen below in Figure 51, will be displayed.



```
      **** Available Plots ****

      1.  Rotor Azimuth Angle v. Cyclic Pitch
      2.  Blade Azimuth Position v. Airload
      3.  Rotor Airload Distribution
      4.  Blade Azimuth Position v. Lift
      5.  Rotor Lift Distribution
      6.  Angle of Attack Distribution
      7.  Rotor Tip Path Plane Stall pattern and Thrust Distribution
      8.  Rotor Trim Lissajous Figure
      9.  Lift Distribution (Contour Plot)
      10. CL Distribution (Contour Plot)
      11. Mach Number Distribution (Contour Plot)


Please choose a plot by entering 1-11.
```

Figure 51. Available Plots Page.

Selected plots will be displayed in new windows labeled with the appropriate figure numbers. These plots may be simply viewed on the screen or printed by choosing the appropriate MATLAB menu bar commands. If they are viewed on screen it is recommended that you maximize the window for best clarity and plot resolution. When printing, set your printer to landscape mode for the best results. After you have completed viewing the selected figure, either close or minimize the window. A message will be in the command window asking if you want to view another plot. Answer 'yes' or 'no'. If you chose 'yes' simply repeat the process until you have viewed all required plots. If you choose to view plots 2, 4, or 7, you must manually position the plot titles. To do this, simply use the mouse to position the cursor at the point where you would like the title to begin and click the left button. Examples of each plot in Figures 52-62.

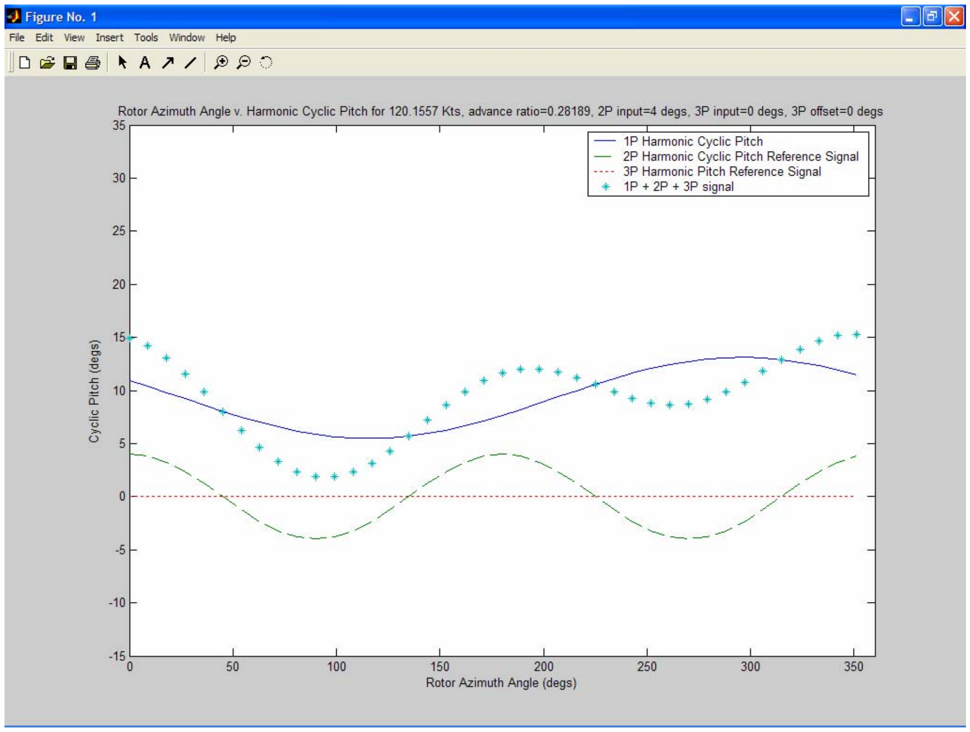
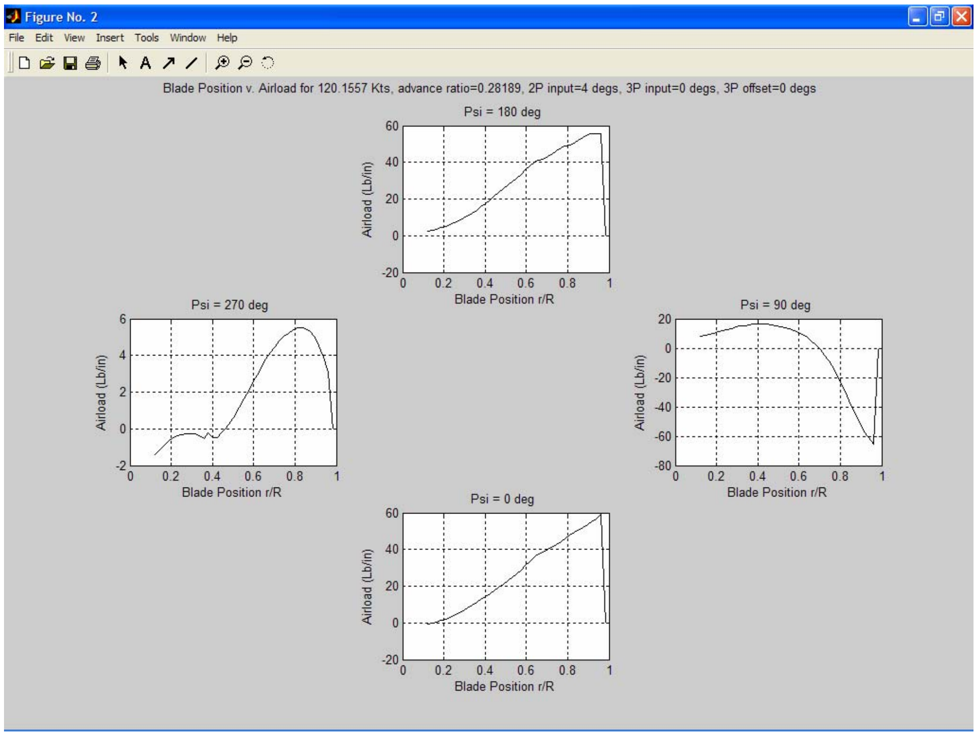
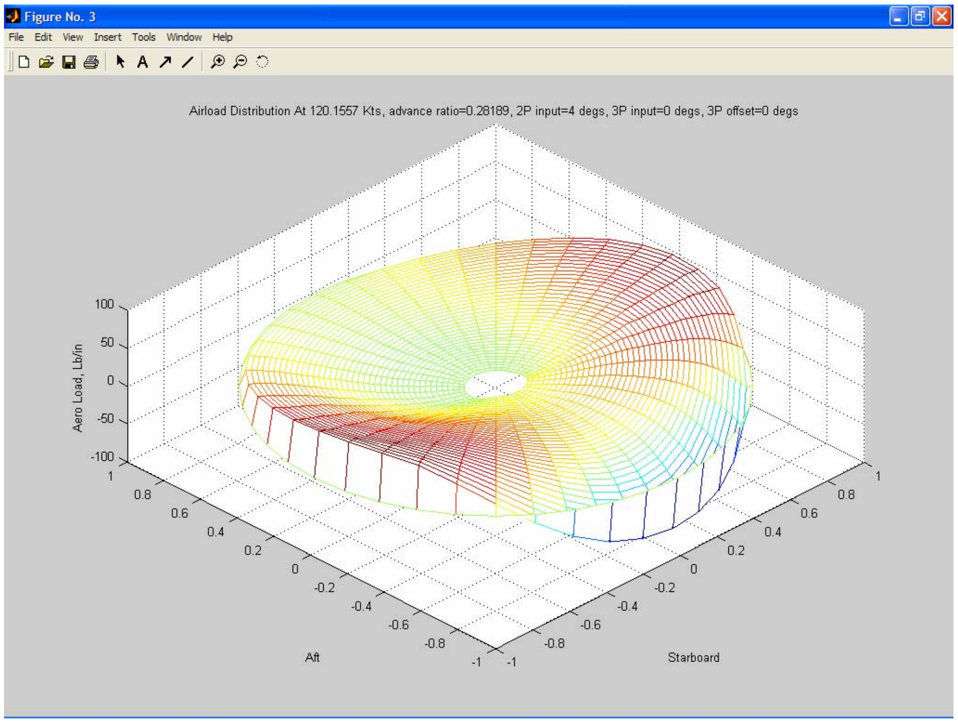


Figure 52. Cyclic Pitch versus Rotor Azimuth Angle.



Blade Position v. Airload for 120.1557 Kts, advance ratio=0.28189, 2P input=4 degs, 3P input=0 degs, 3P offset=0 degs

Figure 53. Blade Position versus Air Load.



Airload Distribution At 120.1557 Kts, advance ratio=0.28189, 2P input=4 degs, 3P input=0 degs, 3P offset=0 degs

Figure 54. Air Load Distribution.

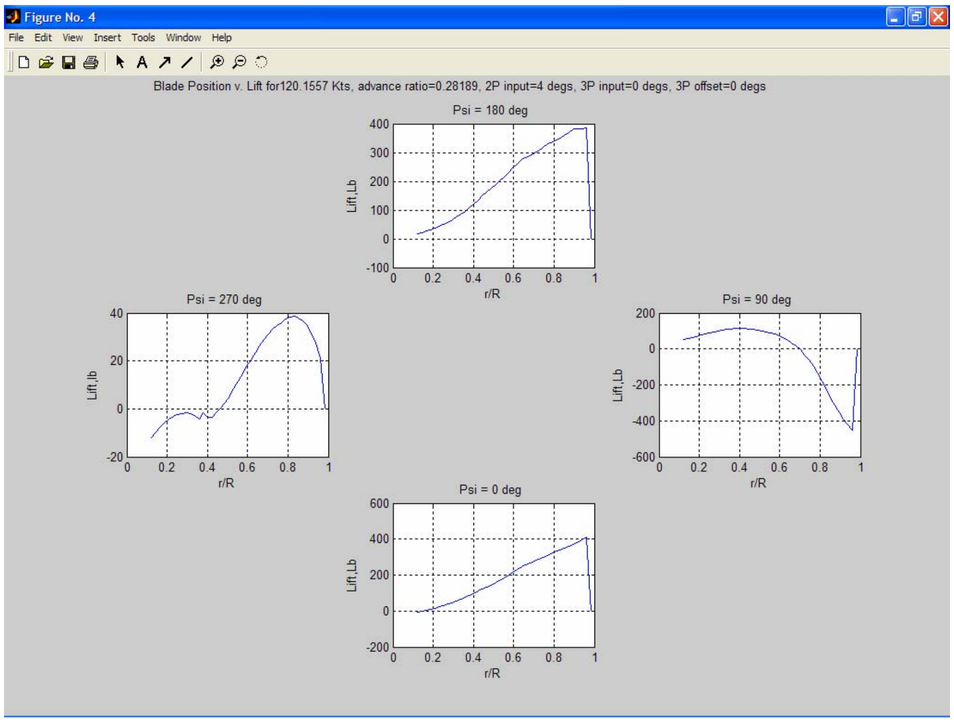


Figure 55. Blade Position versus Lift.

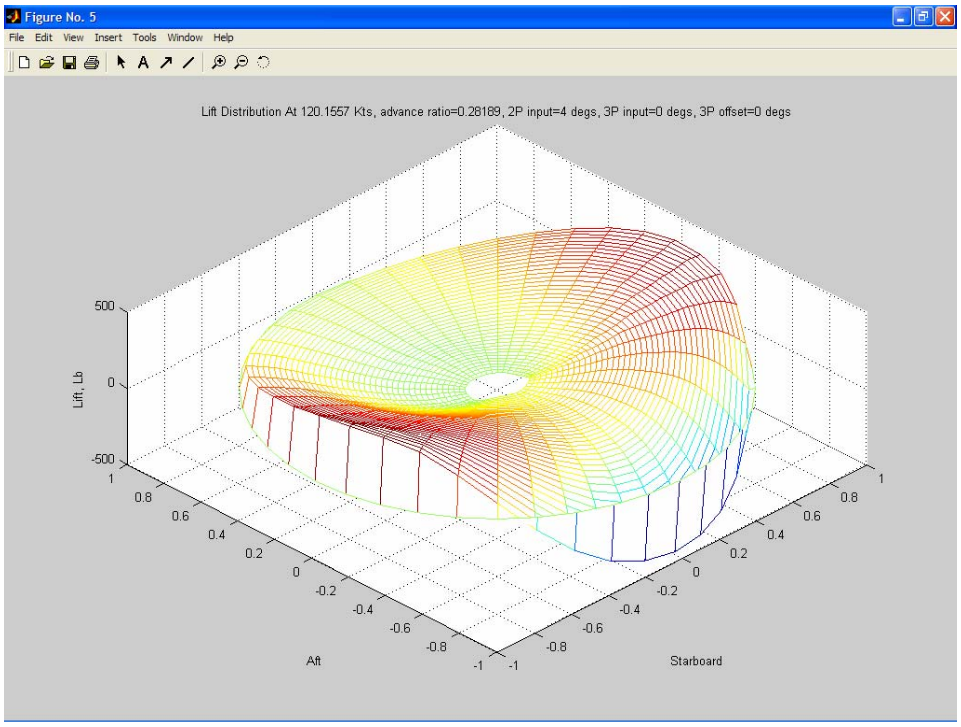


Figure 56. Lift Distribution.

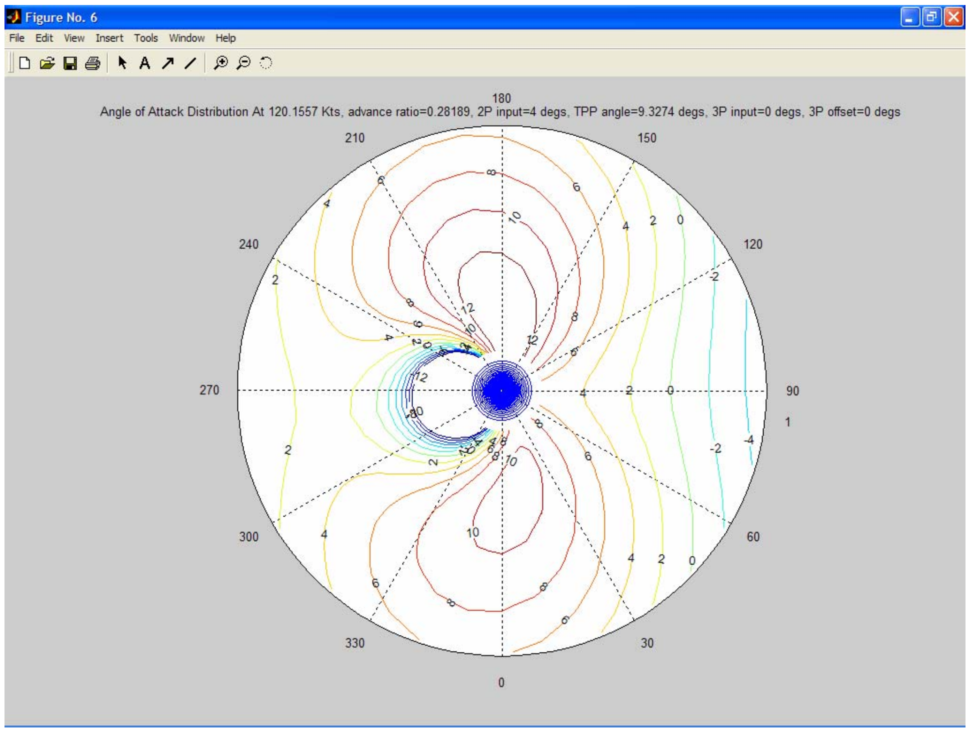


Figure 57. Angle of Attack Distribution.

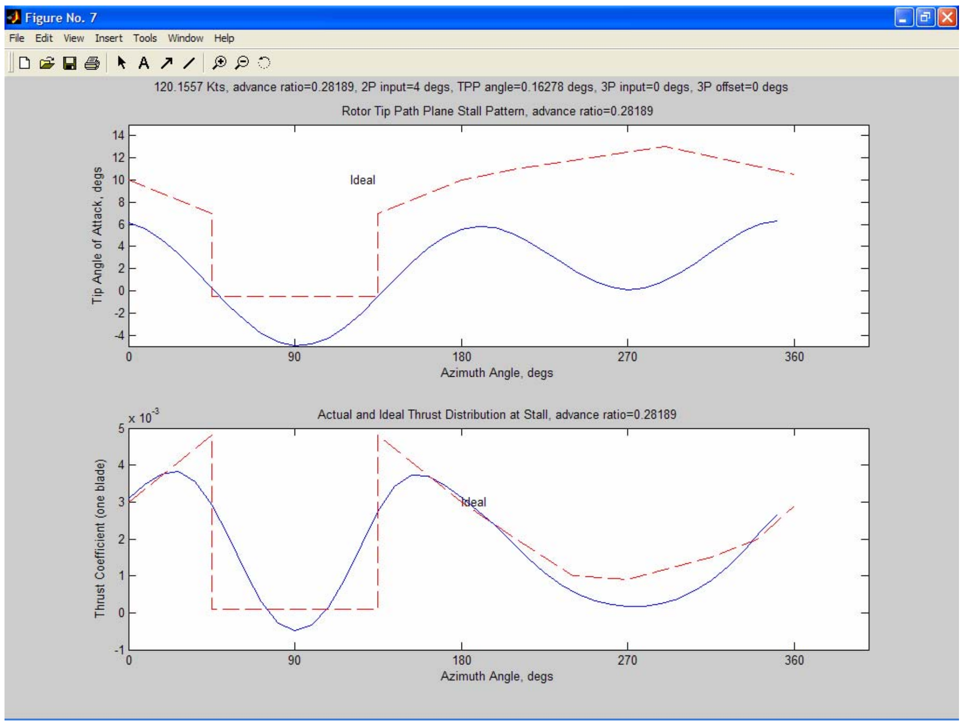


Figure 58. Actual Versus Ideal Rotor Tip Path Plane Angle and Thrust Distribution.

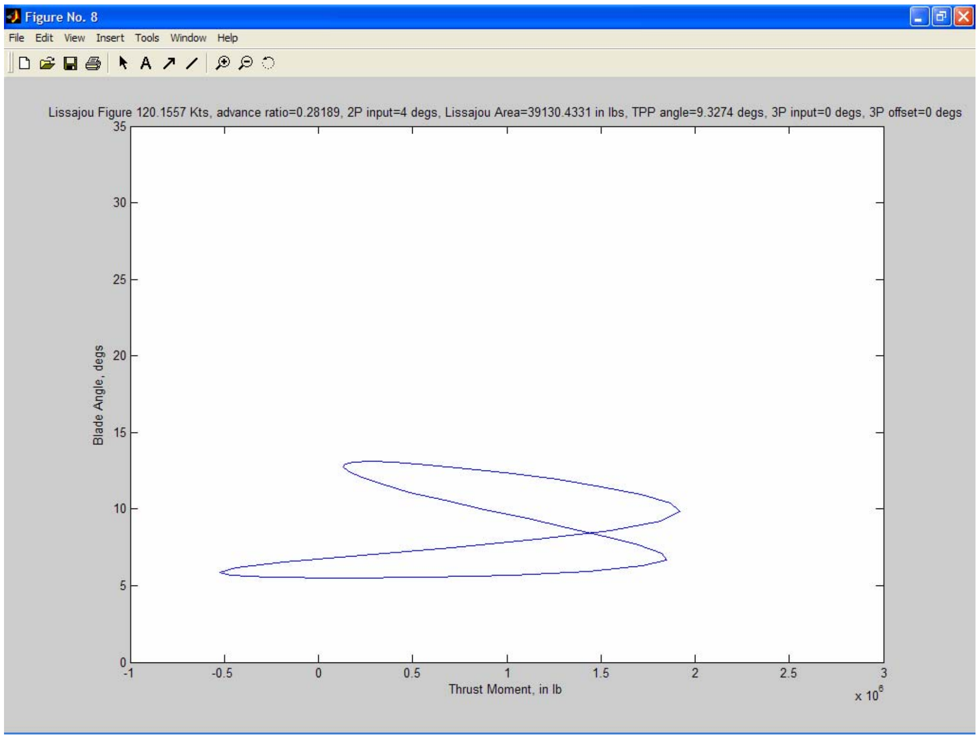


Figure No. 8

Lissajou Figure 120.1557 Kts, advance ratio=0.28189, 2P input=4 degs, Lissajou Area=39130.4331 in lbs, TPP angle=9.3274 degs, 3P input=0 degs, 3P offset=0 degs

Figure 59. Lissajou Figure.

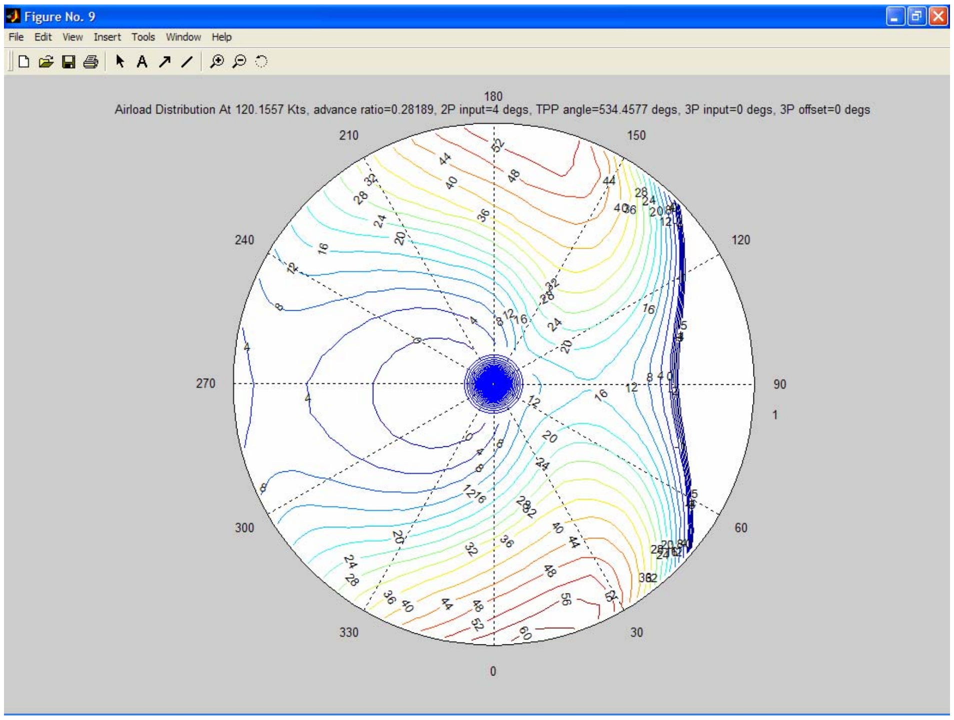


Figure No. 9

Airload Distribution At 120.1557 Kts, advance ratio=0.28189, 2P input=4 degs, TPP angle=534.4577 degs, 3P input=0 degs, 3P offset=0 degs

Figure 60. Air Load Contour Plot.

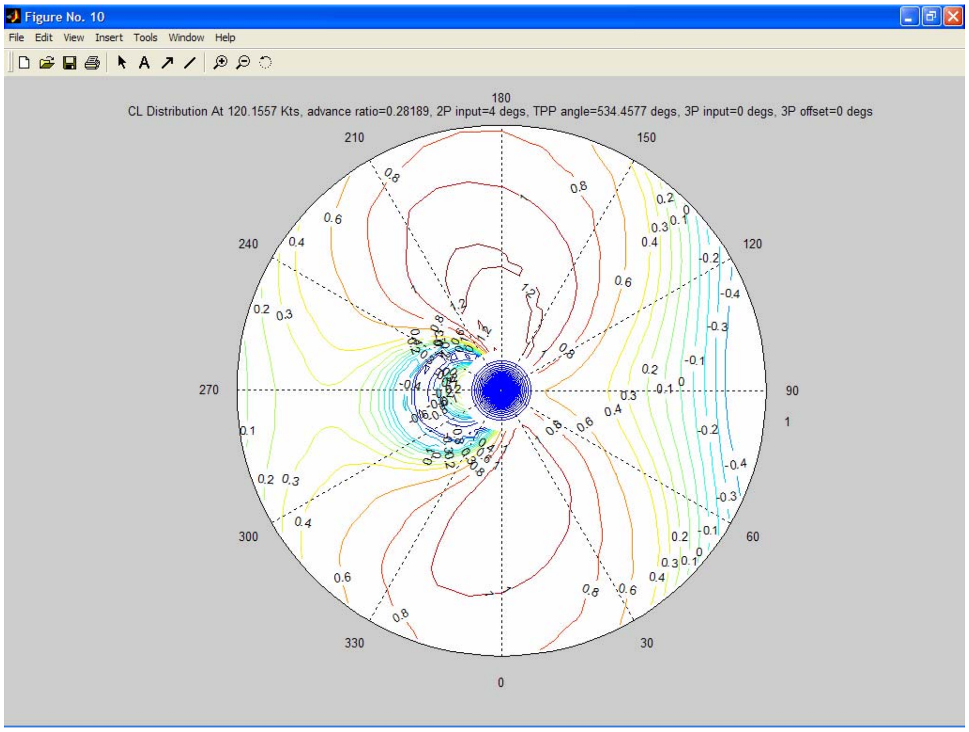


Figure 61. C_L Distribution Contour Plot.

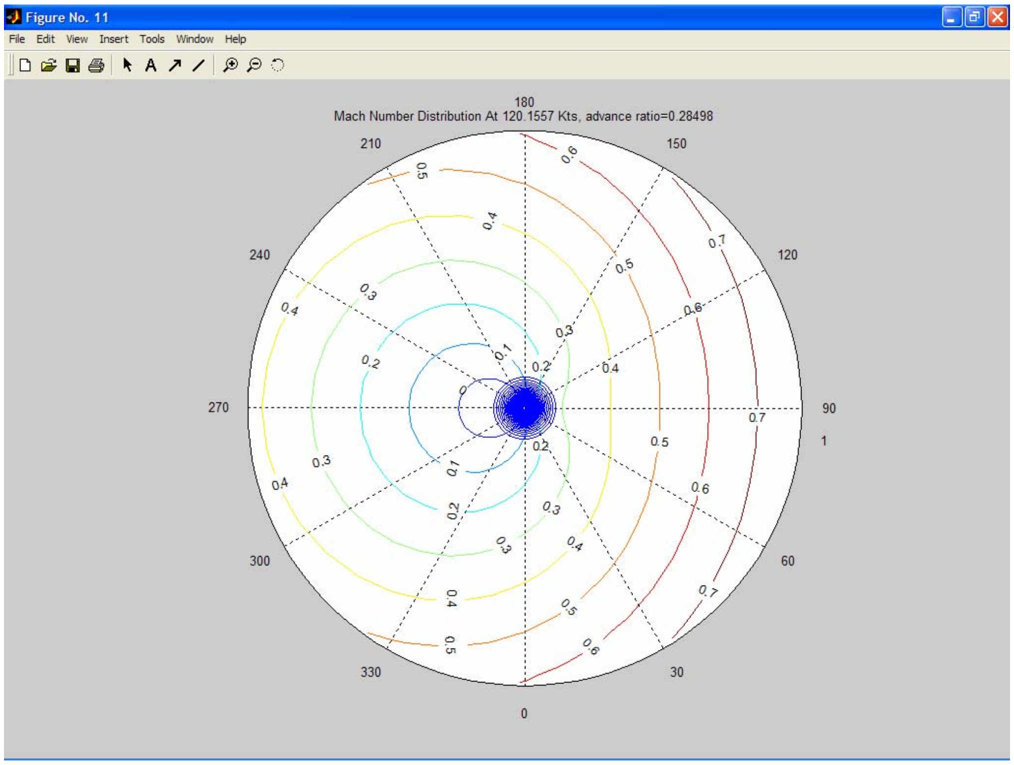
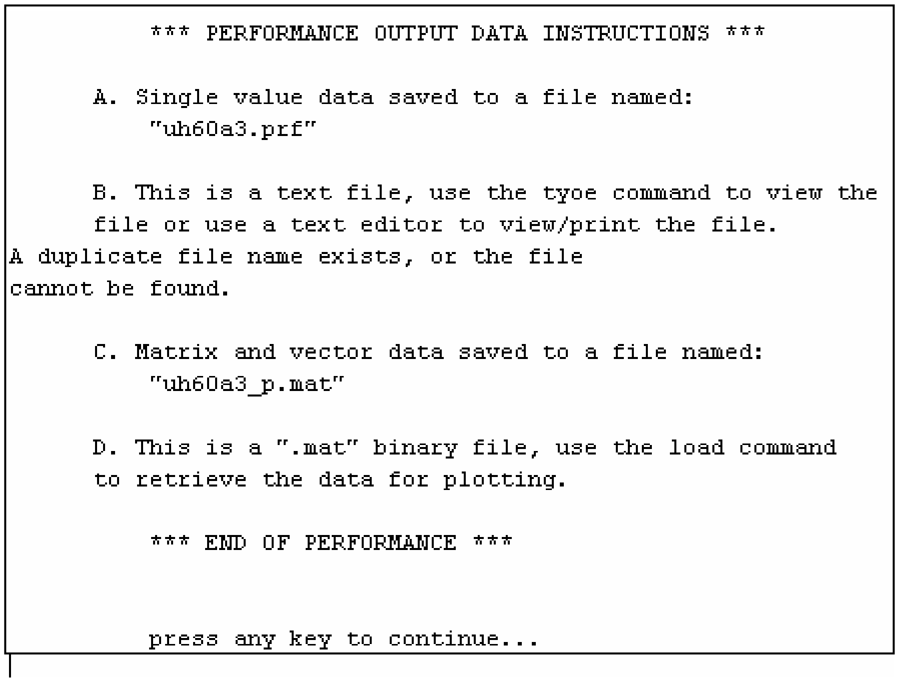


Figure 62. Mach Number Distribution.

If you chose ‘no’, the Performance Output Data Instructions page will be displayed. See Figure 63.



```
          *** PERFORMANCE OUTPUT DATA INSTRUCTIONS ***

      A. Single value data saved to a file named:
          "uh60a3.prf"

      B. This is a text file, use the tyoe command to view the
      file or use a text editor to view/print the file.
A duplicate file name exists, or the file
cannot be found.

      C. Matrix and vector data saved to a file named:
          "uh60a3_p.mat"

      D. This is a ".mat" binary file, use the load command
      to retrieve the data for plotting.

          *** END OF PERFORMANCE ***


          press any key to continue...
```

Figure 63. Performance Output Data Instructions Page.

3. Saved Data

The vector and matrix data generated by the rotor performance module of JANRAD 2P_RVR is saved to the current directory with the file name you entered at the Save Instruction page and concatenated with a ‘_p’ and a ‘.mat’ extension. The following vectors and matrices are saved.

1. *r* – a vector containing the radii (in feet) to the center of each blade element. This is a row vector containing n elements, where n equals the number of blade elements plus one.

2. ψ - a vector containing the azimuth angles (in degrees) around the rotor disk. This is a column vector containing n elements, where n equals the number of azimuth sectors.

3. v_i – a vector containing the induced velocity distribution (in ft/sec)along the rotor blade. This is a row vector containing n elements, where n equals the number of blade elements plus one.

4. θ - a vector containing the 1/rev cyclic pitch (in degrees) with respect to azimuth angle at 0.7 r/R. This is a column vector containing n entries, where n equals the number of azimuth sectors.

5. θ_{xp} – a vector containing the total cyclic pitch, 1/rev + 2/rev +3/rev, (in degrees) with respect to azimuth angle at 0.7 r/R. This is a column vector containing n entries, where n equals the number of azimuth sectors.

6. β_t – a vector containing blade pitch (in degrees) along the rotor blade. This is a row vector containing n elements, where n equals the number of blade elements plus one.

7. α - a matrix containing the angle of attack (in degrees) distribution. The matrix is with m rows and n columns, where m equals the number of azimuth sectors and n equals the number of blade elements plus one.

8. T_ψ - a vector containing the total thrust (in pounds) at each azimuth sector. This is a column vector containing n elements, where n equals the number of azimuth sectors.

9. M_ψ – a vector containing the total thrust moment (in ft-lbs) at each azimuth sector. This is a column vector containing n elements, where n equals the number of azimuth sectors.

10. DM_ψ - a vector containing the total drag moment (in ft-lbs) at each azimuth sector. This is a column vector containing n elements, where n equals the number of azimuth sectors.

11. *dT*- a matrix containing the differential thrust (in pounds) distribution. This is a matrix with m rows and n columns, where m equals the number of azimuth sectors and n equals the number of blade elements plus one.

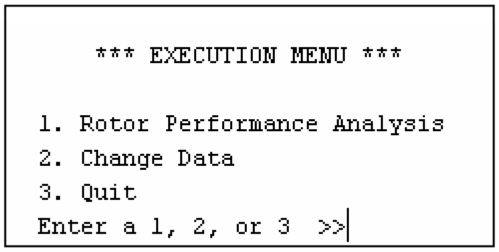
12. *dM* – a matrix containing the differential thrust moment (in ft-lbs) distribution. This is a matrix with m rows and n columns, where m equals the number of azimuth sectors and n equals the number of blade elements plus one.

13. *dD* – a matrix containing the differential drag (in pounds) distribution. This is a matrix with m rows and n columns, where m equals the number of azimuth sectors and n equals the number of blade elements plus one.

14. *$C_L$plot* – a matrix containing the CL distribution over the rotor disk. This is a matrix with n rows and n columns, where n equals the number of blade elements.

15. *$C_D$plot* – a matrix containing the CD distribution over the rotor disk. This is a matrix with n rows and n columns, where n equals the number of blade elements.

When JANRAD 2P_RVR has completed saving the data, the Execution Menu page will be displayed. See Figure 64. Choose the desired function.



```
      *** EXECUTION MENU ***

 1. Rotor Performance Analysis
 2. Change Data
 3. Quit
 Enter a 1, 2, or 3  >>
```

Figure 64. Execution Menu Page.

If you chose option 2, you will be able to reload the file you were just working on or load an entirely different file. If you chose to load an entirely different file, it must be of the same type (either Conventional or Compound Helicopter) as the one you just completed working on. If you need to change the type of helicopter (conventional or compound) you must select option 3 from the Execution Menu page and restart JANRAD 2P_RVR.

VI. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

The original intent of this investigation was to determine to what extent helicopter forward flight could be improved using Reverse Velocity Rotor (RVR) technology and Higher Harmonic Stall Control (HHSC). JANRAD 2P_RVR computational analysis shows that HHSC alone can produce increases in forward flight speeds of up to ten percent in conventional helicopters. Although these improvements were not as radical as Arcidiacono predicted they are still significant. The redistribution of the lift over the rotor disk allows the helicopter to operate with a greater percentage of the retreating side of the rotor system engulfed in reverse flow while still meeting the rigid trim criteria imposed by the program's original trim module. Employment of the 2/rev feathering incurred minimal power penalties and allowed the blade thrust coefficient to approach ideal values. Use of the HHSC worked equally well when used with compound helicopter configurations up to 175 knots, although the extra weight and drag caused by the wing increased power requirements.

At typical helicopter airspeeds (less than 160 knots) the RVR is somewhat inefficient. The RVR's basic shape does not perform nearly as well as the exotic blades currently in use by rotorcraft but, the use of a symmetrical, double ended airfoil allows the generation of lift with both forward and reverse flows across the blades, which would allow the generation of lift on the retreating side at a dramatically wider range of airspeeds than conventionally shaped blades. RVR equipped rotorcraft would require more power to hover and climb than a rotorcraft equipped with proprietary non-linear twist or taper. The true potential of the RVR is best exploited at high forward flight speeds. As flight speed increases the rotor must be slowed based on predetermined optimum advance ratio scheduling. As demonstrated in the brief JVHL study, the RVR is most efficiently placed in this environment when it is used on an aircraft equipped with a compound wing and auxiliary thrust devices. An RVR equipped aircraft would spend the majority of its time in cruise flight, minimizing time spent hovering and at low airspeeds. At cruise speed the RVR must be controlled by a unique RVR specific 2/rev HHSC control system. This system builds upon the conventional 2/rev HHSC system and is required to control the angle of attack of the blades around the rotor system's azimuth to maximize the RVR's unique performance. This angle of attack manipulation allows for lift generation in all quadrants of the system at cruise speed. The predictable lift generation allows for controlled aerodynamic flight of the rotor disk.

B. RECOMMENDATIONS

Improvements to the program can be made in several ways. First, the graphical user interface features within MATLAB should be harnessed to allow users to enter data via editable dialogue and text boxes. The program can also be modified to allow for offsetting the center of gravity from the center of the rotor hub during the trimming process. As more data are made available more airfoils should be added to the selection list to allow for broader experimentation.

Although JANRAD 2P_RVR has the capacity to accommodate 3/rev inputs, the study was limited to 2/rev inputs. Investigation into the effects of combining 3/rev inputs with both a non-RVR and RVR system may yield unforeseen benefits or behaviors.

The development of a HHSC/RVR blade dynamics and stability and control module should be considered. The module should include provisions for both analytical and visual output. These modules should also be able to 'plug-in' to the existing JANRAD 2P_RVR performance module to give the user a complete preliminary design tool.

With the advent of new V/STOL technologies the use of lift fans, direct jet thrusters, and other lift augmentation devices will become more commonplace. Future versions of the program must be able to fully incorporate these devices and should be able to integrate their effects with the traditional helicopter lift (main rotor) and anti-torque (tail rotor) devices. New techniques will have to be developed to combine the effects of these systems to accurately determine downwash velocities, climb performance, and controllability.

APPENDIX A. JANRAD.M

```
% Joint Army Navy Rotorcraft Analysis and Design
% JANRAD
% Version 2P_RVR
% All modification to original code written by
% MAJ Steven Van Riper
% August 2003
%
% Core programming technology
% developed by
% E. R. Wood, Ph.D.
% MAJ Bob Nicholoson
% MAJ Walter Wirth
% September 1993
%
% This program is an interactive preliminary design tool
% developed to aid the design student in determination of
% initial rotorcraft configurations and in the calculation
% of performance and other parameters with or without
% 1P + 2P + 3P rotor control.
% The program will work for conventional or compound rotorcraft.
% The program can take into account various types of powered lift
% including lift fans and direct jet thrusters.
% It will provide accurate data for airspeeds less than 10
% knots and greater than or equal to 50 knots.

% Variable list for janrad.m, trim.m, thrcalc.m, tmcalc.m, dmcalc.m,
% hh02clcd.m, vr12clcd.m, plotter.m, and perf.m

% a           lift curve slope of rotor system airfoil
% Adisk       area of rotor disk
% Afh         fuselage equivalent flat plate drag area
% Afv         vertical projected area (fuselage area under disk)
% airfoil     rotor system airfoil type (HH02/VR12)
% alpha       angle of attack, rotor blade radial segment
% alpham      matrix of alpha values for each blade element
% alphaT      rotor tip path plane angle
% alphaTfix   determines if user fixes alphaT to specified value
% area        area of Lissajous figures in plotter.m
% b           number of rotor blades
% B           tip loss parameter
% betao       rotor coning angle
% betat       geometric angle, rotor blade radial segment
% bhoriz      span, horizontal tail
% bvert       span, vertical tail
% bwing       span, wing
% cblade      chord, rotor blade
% CD          drag coefficient, rotor blade radial segment
% CDohoriz    profile drag coefficient, horizontal tail
% CDovert     profile drag coefficient, vertical tail
% CD0wing     profile drag coefficient, wing
% CDhoriz     drag coefficient, horizontal tail
% CDplot      matrix of CD values for each blade element
% CDvert       drag coefficient, vertical tail
% CDwing       drag coefficient, wing
% CH           rotor H force coefficient
% CH_sig       CH/solidity
% CL           lift coefficient, rotor blade radial segment
% CLhoriz      lift coefficient, horizontal tail
% CLplot       matrix of CL values for each blade element
% CLvert       lift coefficient, vertical tail
% CLwing       lift coefficient, wing
% CQ           rotor torque coefficient
% CQ_sig       CQ/solidity
% CT           rotor thrust coeeficient
% CT_sig       CT/solidity
% dD           differential drag, rotor blade radial segment
% ddD          differential drag, rotor blade tip
% ddDM         differential drag moment, rotor blade tip
% ddM          differential thrust moment, rotor blade tip
% ddT          differential thrust, rotor blade tip
% delM         change in total thrust moment
% devAC        deviation used within Adjusting Collective routine
% devCY        deviation used within Adjusting Cyclic routine
% devMT        deviation used within Mean Thrust routine
% devTC        deviation used within Trimming Collective routine
% Dftotal      resultant of fuselage drag and aux thrust
% Dfuse        total drag generated by no rotor bodies
% DL           disk loading
% dM           differentail thrust moment, rotor blade radial segment
% DMpsi        total blade drag moment at specific azimuth angle
% dr           rotor blade radial segment width
% Drotor       rotor system drag
% dT           differential thrust, rotor blade radial segment
% Dtplot       matrix containing dTs for all blade elements
% Dhoriz       drag, horizontal tail
% dthetadM     change in cyclic pithch with change in thrust moment
% Dvert        drag, vertical tail
% Dwing        drag, wing
% e            effective hinge offset
% ewing        wing efficiency factor
% filename     name of input file
% FM           figure of merit
% grip         lenght of inner non-aerodynamic protion of blade
% GW           aircraft gross wieght
% helochoice   determines choice of compound or conventional helo
% Hrotor       rotor H force
% hub          gives size of hub in plotter.m
% lamdaT       forward flight induced velocity parameter
% Lftotal      total lift generated by non-rotor bodies
% Lhoriz       lift, horizontal tail
% lifterror    difference between actual lift and desired lift
% LoverT       Lift over thrust percentage
% Lvert        lift, vertical tail
% Lwing        lift, wing
% M1c          first harmonic (cosine) thrust moment coefficient
% M1s          first harmonic (sine) thrust moment coefficient
% Machtip      Mach number at rotor blade tip
% mblade       mass of rotor blade
% Mpsi          total blade thrust moment at specific azimuth angle
% mu            advance ratio
% muchoice      determines if user would like JANRAD to change mu to
%               a desired value
% muerror       difference between actual mu and desired mu
% naz           number of azimuth sectors
% nbe           number od blade elements
% omega         rotor rotational velocity
% PA            pressure altitude
% phi           inflow angle, rotor blade radial segment
% phitip        inflow angle, rotor blade tip
% plotchoice    determine which plot to display in plotter.m
% Protor        power required by the rotor
% psi           azimuth angle
% q             dynamic pressure
% Qrotor        rotor torque
% r             radius, rotor blade radial segment
% R             rotor blade radius
% Rbar          Reff-e
% RbarT         rT*Rbar
% Reff          effective rotor radius, tip loss
% rho           ambient air density
% rT            location of resultant thrust vector
% rvrCP         determines if normal or rvr cyclic pitch scheduling is used
% solidity      solidity
% Shoriz        area, horizontal tail
% Svert         area, vertical tail
% Swing         area, wing
% T             rotor thrust
% Taux          auxiliary thrust
% Tauxchoice    determines if user would like JANRAD to match Taux
%               requirements
% temp          ambient air temperature
% theta         cyclic pitch
% thetahub      hub thetas in plotter.m
% theta1c       first harmonic (cosine) of cyclic pitch
% thetals       first harmonic (sine) of cyclic pitch
% theta2c       second harmonic (cosine) of cyclic pitch
% theta3s       third harmonic (sin) of cyclic pitch
% theta3Poffset 3/rev offset
% thetao        collective pitch at .7 r/R
% thetaXP       total vlue of cyclic pitch with all inputs
% Tpsi          total blade thrust at specific azimuth angle
% twist         geometric rotor blade twist
% Up            vertical component of velocity
% Uptip         vertical component of velocity at tip
% Ut            horizontal component of velocity
% Uttip         horizontal component of velocity at tip
% vertaux       vertical auxiliary thrust in pounds
% vertauxchoice determines if auxiliary vertical thrust devices are
%               installed on the aircraft
% vi            induced velocity
% Vinf          forward airspeed
% Vtip          tip speed
% wblade        wieght of rotor blade
% wingchoice    determines if user wnats to allow JANRAD to resize the wing
% wpercent      percent of aircraft gross wieght to be carried by the wing

% clearing all the variables is the MATLAB environment
clear all
clc
disp (' ')
disp (' ')
disp                                                                                    ('
*********************************************************************')
disp ('    *                                                         *')
disp ('    *      Joint Army/Navy Rotorcraft Analysis and Design                *')
disp ('    *                       (JANRAD)                             *')
disp ('    *                                                         *')
disp ('    *                    Version 2P_RVR                             *')
disp ('    *                     October 2003                            *')
disp ('    *                                                         *')
disp                                                                                    ('
*********************************************************************')
disp (' ')
%
% Determine if the user wants to analyze conventional or compound
% helicopter models
%
flag=1;
while flag > 0
    disp('')
    disp('Do you want to analyze conventional or compound rotorcraft? ')
    disp('')
    helochoice=input('          1. conventional or 2. compound >>');
    while isempty(helochoice),
       disp(' ')
       disp('You must enter a 1 or 2')
       disp('Do you want to analyze conventional or compound rotorcraft? ')
       helochoice=input('1. conventional or 2. compound >>');
    end
    if helochoice~=1 & helochoice~=2
       disp(' ')
       disp(' *** Enter a 1 or 2 ***')
    else
       flag=0;
    end
end

clc
check1=1;
while check1 > 0
   check1=1;
   disp(' ')
   disp('Do you want to edit any existing file or create a new one?')
   disp('')
   check=1;
   while check > 0
      answer=input('      1. edit existing file    2. create new file  >>');


%   *** If editing an existing file: get file name, display edit
%      menu, allow changes to selected variables, and save under
%      desired file name.  Loads to and saves from current
%      directory a a .mat file. ***

        if answer==1
           clc
           disp(' ')
           disp(' ')
           disp('                 *** LOAD INSTRUCTIONS ***')
           disp(' ')
           disp('         A. Input the name of the file to edit.')
           disp('         B. The file was saved in your previous session')
           disp('            with a ".mat" extension.')
           disp('         C. Do not include the extension or quaotations.')
           disp(' ')
           disp('         ex: dsgn1')
           flag=0;
           while flag < 1
              filename1=input('           name of input file: ','s');
              eval(['flag=exist(''',filename1,'.mat'');'])
              if flag < 1
                 disp (' ')
                 disp ('         The file does not exist, try again or <Ctrl-C>')
                 disp ('         to exit program.')
              end
           end
           eval(['load ',filename1])
           while check > 0
              if helochoice==1
                 clc
                 disp(' ')
                 disp('       *** CONVENTIONAL HELICOPTER EDIT MENU ***')
                 disp(' ')
                 disp('        1. pressure altitude        2. temperature')
                 disp('        3. airspeed                4. gross weight')
                 disp('        5. number of blades         6. blade radius')
                 disp('        7. blade chord              8. hinge offset')
                 disp('        9. blade grip length        10. blade twist')
                 disp('        11. blade wieght            12. number of blade elements')
                 disp('        13. rotational velocity     14. # azimuth sectors')
                 disp('        15. lift curve slope        16. airfoil')
                 disp('        17. collective pitch        18. flatplate area')
                 disp('        19. vert projected area     20. horizontal tail area')
                 disp('        21. horizontal tail span    22. horizontal tail CL')
                 disp('        23. horizontal tail CDo     24. vertical tail area')
                 disp('        25. vertical tail span      26. vertical tail CL')
                 disp('        27. vertical tail CDo       28. 2P input (cosine)')
                 disp('        29. 3P input                ')
                 disp('        0. NO CHANGES')
                 alphaTfix=0;itercount=0;helochoice=1;rvrCP=2;
              else
                 clc
                 disp(' ')
                 disp('           *** COMPOUND HELICOPTER EDIT MENU ***')
                 disp(' ')
                 disp('        1. pressure altitude        2. temperature')
```

```
    disp('        3. airspeed               4. gross weight')
    disp('        5. number of blades        6. blade radius')
    disp('        7. blade chord             8. hinge offset')
    disp('        9. blade grip length       10. blade twist')
    disp('        11. blade wieght           12. number of blade elements')
    disp('        13. rotational velocity    14. # azimuth sectors')
    disp('        15. lift curve slope       16. airfoil')
    disp('        17. collective pitch       18. flatplate area')
    disp('        19. vert projected area    20. horizontal tail area')
    disp('        21. horizontal tail span   22. horizontal tail CL')
    disp('        23. horizontal tail CDo    24. vertical tail area')
    disp('        25. vertical tail span     26. vertical tail CL')
    disp('        27. vertical tail CDo      28. 2P input')
    disp('        29. 3P input               30. auxiliary thrust')
    disp('        31. wing area              32. wing span')
    disp('        33. wing CL                34. wing CDo')
    disp('        35. wing efficiency factor')
    disp('        0. NO CHANGES')
    alphaTfix=0;itercount=0;helochoice=2;rvrCP=2;
end
choice=input('      Select the parameter to change: ');
if isempty(choice),
    choice=0;
end
if choice==1,
    clc
    disp(' ')
    PA
    tmp=PA;
    PA=input('Pressure altitude (ft); ');
    if isempty(PA),
        PA=tmp;
    end
elseif choice==2,
    clc
    disp(' ')
    temp
    tmp=temp;
    temp=input('temperature (deg F):');
    if isempty(temp),,
        temp=tmp;
    end
elseif choice==3,
    clc
    disp(' ')
    Vinf=Vinf/1.69
    tmp=Vinf;
    Vinf=input('Airspeed (knots): ')*1.69;
    if isempty(Vinf),
        Vinf=tmp*1.69;
    end
elseif choice==4,
    clc
    disp(' ')
    GW
    tmp=GW;
    GW=input('Aircraft Groos weight (lbs): ');
    if isempty (GW),
        GW=tmp;
    end
elseif choice==5,
    clc
    disp(' ')
    b
    tmp=b;
    b=input('Number of blade: ');
    if isempty(b),
        b=tmp;
    end
elseif choice==6,
    clc
    disp(' ')
    R
    tmp=R;
    R=input('Blade radius; center of hub to blade tip (ft): ');
    if isempty(R),
        R=tmp;
    end
elseif choice==7,
    clc
    disp(' ')
    cblade
    tmp=cblade;
    cblade=input('Blade chord (ft); ');
    if isempty(cblade),
        cblade=tmp;
    end
elseif choice==8,
    clc
    disp(' ')
    e
    tmp=e;
    e=input('Hinge offset (ft); ');
    if isempty(e),
        e=tmp;
    end
elseif choice==9,
    clc
    disp(' ')
    grip
    tmp=grip;
    grip=input('non-aerodyn inboard portion of blade (ft): ');
    if isempty (grip),
        grip=tmp;
    end
    if grip < 1e-1,
        grip=1e-10;
    end
elseif choice==10
    clc
    disp(' ')
    twist=-twist*57.3
```

```
   tmp=twist;
   twist=input('blade twist (deg): ');
   if isempty (twist),
      twist=abs(tmp)/57.3;
   else
      twist=abs(twist)/57.3;
   end
elseif choice==11,
   clc
   disp(' ')
   wblade
   tmp=wblade;
   wblade=input('Weight of aero portion of blade (lbs): ');
   if isempty(wblade),
      wblade=tmp;
   end
elseif choice==12,
   clc
   disp(' ')
   nbe
   tmp=nbe;
   nbe=input('number of blade elements: ');
   if isempty(nbe),
      nbe=tmp;
   end
elseif choice==13,
   clc
   disp(' ')
   omega
   tmp=omega;
   omega=input('Rotor rotational velocity (rad/sec): ');
   if isempty (omega),
      omega=tmp;
   end
elseif choice==14,
   clc
   disp(' ')
   naz
   tmp=naz;
   naz=input('number of azimuth sectors: ');
   if isempty (naz),
      naz=tmp;
   end
elseif choice==15,
   clc
   disp(' ')
   a
   tmp=a;
   a=input('Average lift curve slope (CL vs alpha); ');
   if isempty(a),
      a=tmp;
   end
elseif choice==16,
   clc
   disp('')
   airfoil
```

```
    tmp=airfoil;
    flag=1;
    while flag > 0
        airfoil=input('Airfoil 1. HH-02 2. VR-12 3. SC1095 4. 0012 >>');
        if isempty(airfoil),
            airfoil=tmp;
        end
        if airfoil==1,
            flag=0;
        elseif airfoil==2,
            flag=0;
        elseif airfoil==3,
            flag=0;
        elseif airfoil==4,
            flag=0;
        else
            disp(' ')
            disp(' *** Enter a 1, 2, 3, or 4 *** ')
        end
    end
elseif choice==16 & helochoice==2,
    clc
    disp('')
    airfoil
    tmp=airfoil;
    flag=1;
    while flag > 0
        airfoil=input('Airfoil 1. HH-02 2. VR-12 3. SC1095 4. 0012 5. RVR>>');
        if isempty(airfoil),
            airfoil=tmp;
        end
        if airfoil==1,
            flag=0;
        elseif airfoil==2,
            flag=0;
        elseif airfoil==3,
            flag=0;
        elseif airfoil==4,
            flag=0;
        elseif airfoil==5,
            flag=0;
        else
            disp(' ')
            disp(' *** Enter a 1, 2, 3, 4, or 5 *** ')
        end
    end
elseif choice==17,
    clc
    disp(' ')
    thetao=thetao*57.3
    tmp=thetao;
    thetao=input('collective pitch at .7 r/R (deg): ')/57.3;
    if isempty(thetao),
        thetao=tmp/57.3;
    end
elseif choice==18,
    clc
    disp(' ')
    Afh
    tmp=Afh;
    Afh=input('Aircraft equivalent flatplate area (ft^2):');
    if isempty(Afh),
        Afh=tmp;
    end
elseif choice==19,
    clc
    disp(' ')
    Afv
    tmp=Afv;
    Afv=input('Verical projected area (ft^2):');
    if isempty(Afv),
        Afv=tmp;
    end
elseif choice==20,
    clc
    disp(' ')
    Shoriz
    tmp=Shoriz;
    Shoriz=input('Horizontal tail area (ft^2): ');
    if isempty(Shoriz),
        Shoriz=tmp;
    end
    if Shoriz < 1e-10,
        Shoriz=1e-10;
    end
elseif choice==21,
    clc
    disp(' ')
    bhoriz
    tmp=bhoriz;
    bhoriz=input('Horizontal tail span (ft): ');
    if isempty(bhoriz),
        bhoriz=tmp;
    end
    if bhoriz < 1e-10,
        bhoriz=1e-10;
    end
elseif choice==22,
    clc
    disp(' ')
    CLhoriz
    tmp=CLhoriz;
    CLhoriz=input('Expected CL for the horizontal tail: ');
    if isempty (CLhoriz),
        CLhoriz=tmp;
    end
elseif choice==23,
    clc
    disp(' ')
    CDohoriz
    tmp=CDohoriz;
    CDohoriz=input('Horizontal tail prolfile drag coef (CDo): ');
```

```
    if isempty(CDohoriz),
        CD0horiz=tmp;
    end
elseif choice==24,
    clc
    disp(' ')
    Svert
    tmp=Svert;
    Svert=input('Vertical tail area (ft^2): ');
    if isempty(Svert),
        Svert=tmp;
    end
    if Svert < 1e-10,
        Svert=1e-10;
    end
elseif choice==25,
    clc
    disp(' ')
    bvert
    tmp=bvert;
    bvert=input('Vertical tail span (ft): ');
    if isempty(bvert),
        bvert=tmp;
    end
    if bvert < 1e-10,
        bvert=1e-10;
    end
elseif choice==26
    clc
    disp(' ')
    CLvert
    tmp=CLvert;
    CLvert=input('Expected CL for the vertical tail: ');
    if isempty(CLvert),
        CLvert=tmp;
    end
elseif choice==27,
    clc
    disp(' ')
    CDovert
    tmp=CDovert;
    CDovert=input('Vertical tail profile drag coef (CDo); ');
    if isempty(CDovert),
        CDovert=tmp;
    end
elseif choice==28,
    clc
    disp(' ')
    twoPinput
    tmp=twoPinput;
    twoPinput=input('Longitudinal 2P harmonic input (degs): ');
    if isempty (twoPinput),
        twoPinput=tmp;
    end
elseif choice==29
    clc
    disp('')
    threePinput
    tmp=threePinput;
    threePinput=input('3P harmonic input (degs): ');
    disp('')
    threePoffset
    tmp=threePoffset;
    threePoffset=input('3P input offset, must be between 0 and 119 degs: ');
    if isempty (threePinput)
       threePinput=tmp;
    end
elseif helochoice==2 & choice==30,
    clc
    disp(' ')
    Taux
    tmp=Taux;
    Taux=input('Auxiliary thrust (lbs): ');
    if isempty (Taux),
       Taux=tmp;
    end
elseif helochoice==2 & choice==31,
    clc
    disp(' ')
    Swing
    tmp=Swing;
    Swing=input('Wing Area (ft^2):');
    if isempty (Swing),
       Swing=tmp;
    end
    if Swing < 1e-10
       Swing=1e-10;
    end
elseif helochoice==2 & choice==32,
    clc
    disp(' ')
    bwing
    tmp=bwing;
    bwing=input('Wing span (ft); ');
    if isempty(bwing),
       bwing=tmp;
    end
    if bwing < 1e-10,
       bwing=1e-10;
    end
elseif helochoice==2 & choice==33,
    clc
    disp(' ')
    CLwing
    tmp=CLwing;
    CLwing=input('expected CL for the wing:');
    if isempty(CLwing),
       CLwing=tmp;
    end
elseif helochoice==2 & choice==34,
    clc
    disp(' ')
```

```
                        CDowing
                        tmp=CDowing;
                        CDowing=input('Wing profile drag coef (CDo):');
                        if isempty(CDowing),
                           CDowing=tmp;
                        end
                     elseif helochoice==2 & choice==35,
                        clc
                        disp(' ')
                        ewing
                        tmp=ewing;
                        ewing=input('Wing efficiency factor (e): ');
                        if isempty(ewing),
                           ewing=tmp;
                        end
                        if ewing < 1e-10,
                           ewing=1e-10;
                        end
                     elseif helochoice==2 & choice==0
                        GW1=GW;
                        clc
                        if Vinf > 16.9
                           disp('')
                           disp('Do you want to fix the Tip Path Plane (recommended for compound
helicopters),')
                           alphaTfix=input('1. yes or 2. no?');
                           if alphaTfix==1
                              disp('')
                              alphaTinput=input('Enter the Tip Path Plane angle in degrees: ');
                              alphaTinput=alphaTinput/57.3;
                              disp('***   You have fixed the Tip Path Plane angle, please set auxiliary   ***')
                              disp('***   thrust to equal total drag to avoid erroneous solutions.        ***')
                           else
                              disp('')
                              disp('Tip path plane will be determined by JANRAD')
                              disp('Press any key to continue...')
                              pause
                           end
         % added 14-16 Sep 2003 to allow user to disable the retrim routine, select Taux based
         % on drag, and allow for RVR based cyclic inputs in trim.m
                           flag=1;
                           while flag > 0
                              disp('')
                              disp('Do you want JANRAD to set auxiliary thrust to equal total drag, ')
                              Tauxchoice=input('1. yes or 2. no?');
                              while isempty(Tauxchoice),
                                 disp(' ')
                                 disp('You must enter a 1 or 2')
                                 disp('Do you want JANRAD to set auxiliary thrust to equal total drag, ')
                                 Tauxchoice=input('1. yes or 2. no?');
                              end
                              if Tauxchoice~=1 & Tauxchoice~=2
                                 disp(' ')
                                 disp(' *** Enter a 1 or 2 ***')
                              elseif Tauxchoice==1
                                 disp('JANRAD will adjust auxiliary thrust to equal total drag')
```

```
                                        disp('Press any key to continue...')
                                        pause
                                        flag=0;
                                    elseif Tauxchoice==2
                                        disp('*** Auxiliary thrust defined by user.  ***')
                                        disp('Press any key to continue...')
                                        pause
                                        flag=0;
                                    else
                                        flag=0;
                                    end
                                end
                                rvrCP=2;
                                if Vinf/1.69 > 200
                                    rvrCP=0;
                                    flag=1;
                                    while flag > 0
                                        disp('')
                                        disp('Do you want JANRAD to use RVR based (constrained) 2/rev Cyclic
Pitch, ')
                                        rvrCP=input('1. yes or 2. no?');
                                        while isempty(rvrCP),
                                            disp(' ')
                                            disp('You must enter a 1 or 2')
                                            disp('Do you want JANRAD to use RVR based (constrained) 2/rev Cyclic
Pitch, ')
                                            rvrCP=input('1. yes or 2. no?');
                                        end
                                        if rvrCP~=1 & rvrCP~=2
                                            disp(' ')
                                            disp(' *** Enter a 1 or 2 ***')
                                        elseif rvrCP==1
                                            disp('')
                                            disp('RVR 2/rev Cyclic Pitch will be used')
                                            disp('Press any key to continue...')
                                            pause
                                            flag=0;
                                        elseif rvrCP==2
                                            disp('')
                                            disp('Conventional 2/rev cyclic pitch will be used')
                                            disp('Press any key to continue...')
                                            pause
                                            flag=0;
                                        else
                                            flag=0;
                                        end
                                    end
                                else
                                end
                            else
                            end
                            clc
                            disp(' ')
                            disp(' ')
                            disp('            *** SAVE INSTRUCTIONS ***')
                            disp(' ')
```

```
      disp('          A.  Save the new data to a specificed file name.')
      disp('          B.  Do not use an extension or quotations.')
      disp('          C.  Use the letter/number combinations of 20 characters or less.')
      disp('          D.  The file will be saved with a ".mat" extension.')
      disp(' ')
      disp('          ex: dsgn_2')
      disp(' ')
      disp('          E.  If you made no changes, press < Enter >, the file will')
      disp('              be saved with the original name.')
      flag=1;
      while flag > 0
         filename0=filename1;
         filename1=input('       save file as: ','s');
         if isempty(filename1)
            filename1=filename0;
         end
         clear filename0
         if length(filename1) > 20,
            disp(' ')
            disp('          use 20 characters or less')
            flag=1;
         else
            flag=0;
         end
      end
         eval(['save ',filename1])
         check=0;
   elseif choice==0
      if helochoice==1
         Swing=1e-10;
         bwing=1e-10;
         CLwing=0;
         CDowing=0;
         ewing=1e-10;
         Taux=0;
         GW1=GW;
      else
      end
      clc
      disp(' ')
      disp(' ')
      disp('              *** SAVE INSTRUCTIONS ***')
      disp(' ')
      disp('          A.  Save the new data to a specificed file name.')
      disp('          B.  Do not use an extension or quotations.')
      disp('          C.  Use the letter/number combinations of 20 characters or less.')
      disp('          D.  The file will be saved with a ".mat" extension.')
      disp(' ')
      disp('          ex: dsgn_2')
      disp(' ')
      disp('          E.  If you made no changes, press < Enter >, the file will')
      disp('              be saved with the original name.')
      flag=1;
      while flag > 0
         filename0=filename1;
         filename1=input('       save file as: ','s');
```

```
                if isempty(filename1)
                    filename1=filename0;
                end
                clear filename0
                if length(filename1) > 20,
                    disp(' ')
                    disp('          use 20 characters or less')
                    flag=1;
                else
                    flag=0;
                end
            end
                eval(['save ',filename1])
                check=0;
        else
            disp(' ')
            disp('      Enter a displayed number ...press any key to continue')
            pause
        end
    end


%   *** If creating a new file: get input for required variables
%       and save under desired file name.  Save to current
%       directory as a .mat file. ***

  elseif answer==2,
    alphaTfix=0;itercount=0;
    clc
    PA=input('Pressure Altitude (ft): ');
    while isempty(PA),
        disp(' ')
        disp('You must enter a numerical value')
        PA=input('Pressure altitude (ft): ');
    end
    temp=input('Temperature (deg F):');
    while isempty(temp),
        disp(' ')
        disp('You must enter a numerical value')
        temp=input('Temperature (deg F): ');
    end
    Vinf=input('Airspeed (knots): ')*1.69;
    while isempty(Vinf),
        disp(' ')
        disp('You must enter a numerical value')
        Vinf=input('Airspeed (knots): ')*1.69;
    end
    GW=input('Aircraft gross wieght (lbs): ');
    while isempty(GW),
        disp(' ')
        disp('You must enter a numerical value')
        GW=input('Aircraft gross wieght (lbs):');
    end
    b=input('Number of Blades: ');
    while isempty(b),
        disp(' ')
```

```
   disp('You must enter a numerical value')
   b=input('Number of Blades: ');
end
R=input('Blade radius: center of hub to blade tip (ft): ');
while isempty(R),
   disp(' ')
   disp('You must enter a numerical value')
   R=input('Blade radius: center of hub to blade tip (ft): ');
end
cblade=input('Blade chord (ft): ');
while isempty(cblade),
   disp(' ')
   disp('You must enter a numerical value')
   cblade=input('Blade chord (ft): ');
end
e=input('Hinge offset (ft)');
while isempty(e)
   disp(' ')
   disp('You must enter a numercial value')
   e=input('Hinge offset (ft)');
end
grip=input('Non-aerodynamic inboard portion of blade (ft): ');
while isempty(grip)
   disp(' ')
   disp('You must enter a numercial value')
   grip=input('Non-aerodynamic inboard portion of blade (ft): ');
end
while grip < 1e-10,
   grip=1e-10;
end
twist=input('Blade twist (deg): ')/57.3;,twist=abs(twist);
while isempty(twist),
   disp(' ')
   disp('You must enter a numerical value')
   twist=input('Blade twist (deg): ')/57.3;,twist=abs(twist);
end
wblade=input('weight of aero portion of one blade (lbs): ');
while isempty(wblade),
   disp(' ')
   disp('You must enter a numerical value')
   wblade=input('weight of aero portion if one blade (lbs): ');
end
nbe=input('Number of blade elements: ');
while isempty(nbe),
   disp(' ')
   disp('You must enter a numerical value')
   nbe=input('Number of blade elements: ');
end
omega=input('Rotor rotational velocity (rad/sec): ');
while isempty(omega),
   disp(' ')
   disp('You must enter a numerical value')
   omega=input('Rotor rotational velocity (rad/sec): ');
end
naz=input('Number of azimuth sectors: ');
while isempty(naz),
   disp(' ')
   disp('You must enter a numerical value')
   naz=input('Number of azimuth sectors: ');
end
a=input('Lift curve slope of rotor airfoil (CL vs alpha): ');
while isempty(a),
   disp(' ')
   disp('You must enter a numerical value')
   a=input('Lift curve slope of rotor airfoil (CL vs alpha): ');
end
flag=1;
while flag > 0
   if helochoice==1
      airfoil=input('Airfoil 1. HH-02 2. VR-12 3. SC1095 4. 0012 >>');
      while isempty(airfoil),
         disp(' ')
         disp('You must enter a numerical value')
         airfoil=input('Airfoil 1. HH-02 2. VR-12 3. SC1095 4. 0012 >>');
      end
      if airfoil~=1 & airfoil~=2 & airfoil~=3 & airfoil~=4
         disp(' ')
         disp(' *** Enter a 1, 2, 3, or 4 ***')
      else
         flag=0;
      end
   else
      airfoil=input('Airfoil 1. HH-02 2. VR-12 3. SC1095 4. 0012 5. RVR >>');
      while isempty(airfoil),
         disp(' ')
         disp('You must enter a numerical value')
         airfoil=input('Airfoil 1. HH-02 2. VR-12 3. SC1095 4. 0012 5. RVR >>');
      end
      if airfoil~=1 & airfoil~=2 & airfoil~=3 & airfoil~=4 & airfoil~=5
         disp(' ')
         disp(' *** Enter a 1, 2, 3, 4, or 5 ***')
      else
         flag=0;
      end
   end
end
thetao=input('collective pitch at .7 r/R (deg): ')/57.3;
while isempty(thetao),
   disp(' ')
   disp('You must enter a numerical value')
   thetao=input('Collective pitch at .7 r/R (deg): ')/57.3;
end
Afh=input('Aircraft equivalent flatplate area (ft^2):');
while isempty(Afh)
   disp(' ')
   disp('You must enter a numerical value')
   Afh=input('Aircraft equivalent flatplate area (ft^2):');
end
Afv=input('Vertical projected area (ft^2): ');
while isempty(Afv)
   disp(' ')
   disp('Tou must enter a numerical value')
   Afv=input('Vertical projected area (ft^2): ');
end
Shoriz=input('Horizontal tail area, 0 if none (ft^2): ');
while isempty(Shoriz),
   disp(' ')
   disp('You must enter a numerical value')
   Shoriz=input('Horizontal tail area, 0 if none (ft^2): ');
end
if Shoriz~=0,
   bhoriz=input('Horizontal tail span (ft): ');
   while isempty(bhoriz),
      disp(' ')
      disp('You must enter a numerical value')
      bhoriz=input('Horizontal tail span (ft): ');
   end
   if bhoriz < 1e-10,
      bhoriz=1e-10;
   end
   CLhoriz=input('Expected CL for the horizontal tail: ');
   while isempty(CLhoriz),
      disp(' ')
      disp('You must enter a numerical value')
      CLhoriz=input('Expected CL for the horizontal tail: ');
   end
   CDohoriz=input('Horizontal tail profile drag coef (CDo): ');
   while isempty(CDohoriz),
      disp(' ')
      disp('You must enter a numerical value')
      CDohoriz=input('Horizontal tail profile drag coef (CDo): ');
   end
else
   Shoriz=1e-10;
   bhoriz=1e-10;
   CLhoriz=0;
   CDohoriz=0;
end
Svert=input('Vertical tail area, 0 if none (ft^2): ');
while isempty(Svert),
   disp(' ')
   disp('You must enter a numerical value')
   Svert=input('Vertical tail area, 0 if none (ft^2): ');
end
if Svert~=0
   bvert=input('Vertical tail span (ft): ');
   while isempty(bvert),
      disp(' ')
      disp('You must enter a numerical value')
      bvert=input('Vertical tail span (ft): ');
   end
   if bvert < 1e-10,
      bvert=1e-10;
   end
   CLvert=input('Expected CL for the vertical tail: ');
   while isempty(CLvert),
      disp(' ')
      disp('You must enter a numerical value')
```

```
        CLvert=input('Expected CL for the vertical tail: ');
    end
    CDovert=input('Vertical tail profile drag coef (CDo):');
    while isempty(CDovert),
        disp(' ')
        disp('You must enter a numerical value')
        CDovert=input('Vertical tail profile drag coef (CDo):');
    end
else
    Svert=1e-10;
    bvert=1e-10;
    CLvert=0;
    CDovert=0;
end
twoPinput=input('2P longitudinal input (degs): ');
while isempty(twoPinput),
    disp(' ')
    disp('You must enter a numerical value')
    twoPinput=input('2P longitudinal input (degs): ');
end
threePinput=input('3P input (degs): ');
while isempty(threePinput),
    disp(' ')
    disp('You must enter a numerical value')
    threePinput=input('3P input (degs): ');
end
threePoffset=input('3P offset, must be between 0 and 119 degs: ');
while isempty(threePoffset) or threePoffset>119 or threePoffset<0,
    disp('')
    disp('You must enter a numerical value between 0 and 119 degs')
    threePoffset=input('3P offset (degs): ');
end
Swing=1e-10;
bwing=1e-10;
CLwing=0;
CDowing=0;
ewing=1e-10;
Taux=0;
rvrCP=2;
if helochoice==2
    Taux=input('Auxiliary thrust (lbs): ');
    while isempty(Taux),
        disp(' ')
        disp('You must enter a numerical value')
        Taux=input('Auxiliary thrust (lbs): ');
    end
    Swing=input('Wing area, 0 if no wing (ft^2): ');
    while isempty(Swing)
        disp(' ')
        disp('You must enter a numerical value')
        Swing=input('Wing area, 0 if no wing (ft^2): ');
    end
    if Swing~=0,
        bwing=input('Wing span (ft):');
        while isempty(bwing),
            disp(' ')
```

```
                  disp('You must enter a numerical value')
                  bwing=input('Wing span (ft): ');
               end
               if bwing < 1e-10,
                  bwing=1e-10;
               end
               CLwing=input('Expected CL for the wing: ');
               while isempty(CLwing),
                  disp(' ')
                  disp('You must enter a numerical value')
                  CLwing=input('Expected CL for the wing: ');
               end
               CDowing=input('Wing profile drag coef (CDo): ');
               while isempty(CDowing),
                  disp(' ')
                  disp('You must enter a numerical value')
                  CDowing=input('Wing profile drag coef (CDo): ');
               end
               ewing=input('Wing efficiency factor (e): ');
               while isempty(ewing),
                  disp(' ')
                  disp('You must enter a numerical value')
                  ewing=input('Wing efficiency factor (e): ');
               end
               if ewing < 1e-10,
                  ewing=1e-10;
               end
            else
               Swing=1e-10;
               bwing=1e-10;
               CLwing=0;
               CDowing=0;
               ewing=1e-10;
            end
            GW1=GW;
            clc
            if Vinf > 16.9
               disp('')
               disp('Do you want to fix the Tip Path Plane (recommended for compound
helicopters),')
               alphaTfix=input('1. yes or 2. no?');
               if alphaTfix==1
                  disp('')
                  alphaTinput=input('Enter the Tip Path Plane angle in degrees: ');
                  alphaTinput=alphaTinput/57.3;
                  disp('***   You have fixed the Tip Path Plane angle, please set auxiliary   ***')
                  disp('***   thrust to equal total drag to avoid erroneous solutions.        ***')
               else
                  disp('')
                  disp('Tip path plane will be determined by JANRAD')
                  disp('Press any key to continue...')
                  pause
               end
    % added 14-16 Sep 2003 to allow user to disable the retrim routine, select Taux based
    % on drag, and allow for RVR based cyclic inputs in trim.m
               flag=1;
```

```
while flag > 0
   disp('')
   disp('Do you want JANRAD to set auxiliary thrust to equal total drag, ')
   Tauxchoice=input('1. yes or 2. no?');
   while isempty(Tauxchoice),
      disp(' ')
      disp('You must enter a 1 or 2')
      disp('Do you want JANRAD to set auxiliary thrust to equal total drag, ')
      Tauxchoice=input('1. yes or 2. no?');
   end
   if Tauxchoice~=1 & Tauxchoice~=2
      disp(' ')
      disp(' *** Enter a 1 or 2 ***')
   elseif Tauxchoice==1
      disp('JANRAD will adjust auxiliary thrust to equal total drag')
      disp('Press any key to continue...')
      pause
      flag=0;
   elseif Tauxchoice==2
      disp('*** Auxiliary thrust defined by user.  ***')
      disp('Press any key to continue...')
      pause
      flag=0;
   else
      flag=0;
   end
end
rvrCP=2;
if Vinf/1.69 > 200
rvrCP=0;
flag=1;
   while flag > 0
      disp('')
      disp('Do you want JANRAD to use RVR based (constrained) 2/rev Cyclic Pitch,
')

      rvrCP=input('1. yes or 2. no?');
      while isempty(rvrCP),
         disp(' ')
         disp('You must enter a 1 or 2')
         disp('Do you want JANRAD to use RVR based (constrained) 2/rev Cyclic
Pitch, ')

         rvrCP=input('1. yes or 2. no?');
      end
      if rvrCP~=1 & rvrCP~=2
         disp(' ')
         disp(' *** Enter a 1 or 2 ***')
      elseif rvrCP==1
         disp('')
         disp('RVR 2/rev Cyclic Pitch will be used')
         disp('Press any key to continue...')
         pause
         flag=0;
      elseif rvrCP==2
         disp('')
         disp('Conventional 2/rev cyclic pitch will be used')
         disp('Press any key to continue...')
```

```
                  pause
                  flag=0;
                else
                  flag=0;
                end
              end
            else
            end
          else
          end
        else
        end
        clc
        disp(' ')
        disp(' ')
        disp('               *** Save Instructions ***')
        disp(' ')
        disp('        A. Save the data to a specified file name.')
        disp('        B. Do not use an extension or qoutations.')
        disp('        C. Use letter/number combinations of 6 characters or less.')
        disp('        D. The file will be saved with a ".mat" extension.')
        disp(' ')
        disp('        ex: dsgn_1')
        disp(' ')
        disp('        E. If you do not enter a name, the default is "dsgn_1"')
        flag=1;
        while flag > 0
          filename1=input('         save file as: ','s');
          if isempty(filename1)
            filename1='dsgn_1';
          end
          if length(filename1) > 20,
            disp('')
            disp('Use 20 characters or less.')
            flag=1;
          else
            flag=0;
          end
        end
        eval(['save ',filename1])
        check=0;
      else
        disp(' ')
        disp('      Enter a 1 or 2')
      end
    end
    clc
    disp(' ')
    pause(3)
    check2=1;
    while check2 > 0
      clc
      disp(' ')
      disp('              *** EXECUTION MENU ***')
      disp(' ')
      disp('          1. Rotor Performance Analysis')
```

```
        disp('          2. Change Data')
        disp('          3. Quit')
        check3=1;
        while check3 > 0
           answer=input('          Enter a 1, 2, or 3  >>');
           save temp check1 check2 check3 filename1
           if answer==1,
              perf
              load temp
              check3=0;
           elseif answer==2
              clc
              check2=0;
              check3=0;
           elseif answer==3,
              disp(' ')
              disp('          Thank you for using JANRAD')
              check1=0;
              check2=0;
              check3=0;
           end
        end
     end
  end
```

APPENDIX B. DMCALC.M

```
%   dmcalc calculates the total drag along a blade at
%   each azimuth (psi) location

Up=zeros(size(psi*r));
Ut=zeros(size(Up));
alpham=zeros(size(Up));
% **************************
% added for plotting routine
alphamplot=zeros(size(Ut));
% **************************
dD=zeros(size(Up));
ddD=zeros(size(psi));
ddDM=zeros(size(psi));

for i=1:length(psi),
   Up(i,:)=vi.*cos(betao)+Vinf*sin(alphaT)*cos(betao)+Vinf*cos(alphaT)*sin(betao)*cos(psi(i));
   Ut(i,:)=r.*omega+Vinf*cos(alphaT)*sin(psi(i));
   mach = (Vtip.*cos(alphaT).*r./R+Vinf.*sin(psi(i)))/(49.05*sqrt(temp+460));
   phi=atan2(Up(i,:),Ut(i,:));
%   Replace 1P theta with 2P theta
%   alpha=theta(i)+betat-phi;
   alpha=thetaXP(i)+betat-phi;
%   *****************************
%   Test for plotting routine
   alpha1=thetaXP(i)+betat-phi;
   alphamplot(i,:)=alpha1;
%   *****************************
   alpham(i,:)=alpha;

      if airfoil==1,
         [CL,CD]=hh02clcd(alpha);
      % additional airfoils added for experimentation
      elseif airfoil==2,
         [CL,CD]=vr12clcd(alpha);
      elseif airfoil==3,
         [CL,CD]=sc1095r8clcd(alpha,mach);
      elseif airfoil==4,
         [CL,CD]=oo12clcd(alpha,mach);
      elseif airfoil==5,
         [CL,CD]=rvrclcd(alpha,mach);
      end

% Populate CLplot and CDplot matrix
   CLplot(i,:)=CL;
   CDplot(i,:)=CD;

dD(i,:)=0.5*rho*cblade*dr*(Up(i,:).^2).*(CL.*sin(phi)+CD.*cos(phi));
   dDM=dD(i,:).*r;
   DMpsi(i)=sum(dDM);

%   *** calculations for tip loss area ***
```

```
Uptip=Vinf*sin(alphaT)*cos(betao)+Vinf*cos(alphaT)*sin(betao)*cos(psi(i));
   Uttip=(R-(r-Reff)/2)*omega+Vinf*cos(alphaT)*sin(psi(i));
   phitip=atan2(Uptip,Uttip);
   ddD(i)=0.5*rho*cblade*(R-Reff)*(Uptip^2+Uttip.^2)*(.009*cos(phitip'));
   ddDM(i)=ddD(i)*(R-(R-Reff)/2);
   DMpsi(i)=DMpsi(i)+ddDM(i);
end
```

APPENDIX C. PERF.M

```
%   *** Performance output subroutine ***

%   *** Clearing all variable in the MATLAB environment ***

clear

%   *** Loading input and check parameters ***

load temp
eval(['load ',filename1])

trim

disp('trimming complete')

%   *** Calculation of output parameters ***

Protor=mean(DMpsi)*b*omega/550;
Qrotor=mean(DMpsi)*b;
solidity=b*cblade/(pi*R);
CQ=Qrotor/(Adisk*rho*Vtip^2*R);
CH=Hrotor/(Adisk*rho*Vtip^2);
CT_sig=CT/solidity;
CQ_sig=CQ/solidity;
CH_sig=CH/solidity;
Machtip=(Vtip*cos(alphaT)+Vinf)/(49.05*sqrt(temp+460));
   if Vinf < 16.9,
      DL=T/(pi*R^2);
      FM=(T*sqrt(DL/(2*rho)))/(550*Protor);
   else
      DL=0;
      FM=0;
   end

clc
disp(' ')
disp('          *** PERFORMANCE CALCULATIONS COMPLETE ***')
disp(' ')
disp('      Do you want the performance results displayed on screen?')
flag=1;
while flag > 0
   answer=input('      1.  yes     2.  no      >>');
   if answer==2,
      flag=0;
   elseif answer==1

%   *** output to screen ***

clc
disp(' ')
disp('          *** INPUT DATA ***')
eval(['disp("              ',filename1,'")'])
```

```
disp(' ')
fprintf('       Forward velocity    =   %6.0f kts\n',Vinf/1.69)
fprintf('           Temperature     =   %6.0f degs F\n',temp)
fprintf('       Pressure Altitude   =   %6.0f ft\n',PA)
fprintf('           Gross weight    =   %6.0f lbs\n',GW)
fprintf('          Number of blades =   %6.0f \n',b)
fprintf('           Rotor radius    =   %6.2f ft\n',R)
fprintf('           Blade chord     =   %6.2f ft\n',cblade)
fprintf('           Blade twist     =   %6.2f degs\n',-1*twist*57.3)
if airfoil==1
   disp('         Blade airfoil   =   HH-02')
elseif airfoil==2
   disp('         Blade airfoil   =   VR-12')
elseif airfoil==3
   disp('         Blade airfoil   =   SC1095')
elseif airfoil==4
   disp('         Blade airfoil   =   0012')
elseif airfoil==5
   disp('         Blade airfoil   =   RVR')
end
fprintf('   Blade lift curve slope  =   %6.2f \n',a)
fprintf('           Blade wieght    =   %6.2f lbs\n',wblade)
fprintf('     Rotational velocity   =   %6.2f rads/sec\n',omega)
fprintf('       Blade grip length   =   %6.2f ft\n',grip)
fprintf('           Hinge offset    =   %6.2f ft\n',e)
disp(' ')
disp(' ')
disp('press any key to continue...')
pause

clc
disp(' ')
disp('         *** INPUT DATA CONTINUED ***')
eval(['disp(''             ',filename1,''')'])
disp(' ')
fprintf('Equivalent flat plate area =   %6.2f ft^2\n',Afh)
fprintf('  Vertical Projected area  =   %6.2f ft^2\n',Afv)
fprintf('              Wing Area    =   %6.2f ft^2\n',Swing)
fprintf('              Wing Span    =   %6.2f ft\n',bwing)
fprintf('                Wing CL    =   %6.2f \n',CLwing)
fprintf('                Wing CDo   =   %6.4f \n',CDowing)
fprintf('   Wing efficiency factor  =   %6.2f \n',ewing)
fprintf('   Horizontal tail area    =   %6.2f ft^2\n',Shoriz)
fprintf('   Horizontal tail span    =   %6.2f ft\n',bhoriz)
fprintf('   Horizontal tail CL      =   %6.2f \n',CLhoriz)
fprintf('   Horizontal tail CDo     =   %6.4f \n',CDohoriz)
fprintf('   Vertical tail area      =   %6.2f ft^2\n',Svert)
fprintf('   Vertical tail span      =   %6.2f ft\n',bvert)
fprintf('   Vertical tail CL        =   %6.2f \n',CLvert)
fprintf('   Vertical tail CDo       =   %6.4f \n',CDovert)
fprintf('   Auxiliary thrust        =   %6.0f lbs\n',Taux)
fprintf(' Vertical Auxiliary thrust =   %6.0f lbs\n',vertaux)
fprintf('2/rev cyclic input (deg)   =   %6.0f \n',theta2c*57.3)
fprintf('3/rev cyclic input (deg)   =   %6.2f \n',theta3s*57.3)
fprintf('3/rev offset (deg)         =   %6.2f \n',theta3Poffset*57.3)
```

```
disp(' ')
disp(' ')
disp('press any key to continue...')
pause

clc
disp(' ')
disp('         *** CALCULATED DATA ***')
eval(['disp("             ',filename1,'")'])
disp(' ')
fprintf('          Fuselage drag   =   %6.0f lbs\n',Dfuse)
fprintf('             Rotor drag   =   %6.0f lbs\n',Hrotor)
fprintf('              Wing Lift   =   %6.0f lbs\n',Lwing)
fprintf('              Wing Drag   =   %6.0f lbs\n',Dwing)
fprintf('    Horizontal tail lift   =   %6.0f lbs\n',Lhoriz)
fprintf('    Horizontal tail drag   =   %6.0f lbs\n',Dhoriz)
fprintf(' Vertical tail side force  =   %6.0f lbs\n',Lvert)
fprintf('     Vertical tail drag    =   %6.0f lbs\n',Dvert)
fprintf('          Tip path angle  =   %6.2f degs\n',alphaT*57.3)
fprintf('      Rotor Coning angle  =   %6.2f degs\n',betao*57.3)
fprintf('Location of mean thrust(r/R)=  %6.2f \n',rT2)
fprintf('Collective pitch at .7 r/R =   %6.2f degs\n',thetao*57.3)
fprintf('1st lat cyclic term-A1(deg)=   %6.2f \n',theta1c*57.3)
fprintf('1st long cyclic term-B1(deg)=  %6.2f \n',theta1s*57.3)
disp(' ')
disp(' ')
disp('press any key to continue...')
pause

clc
disp(' ')
disp('         *** CALCULATED DATA CONTINUED ***')
eval(['disp("             ',filename1,'")'])
disp(' ')
fprintf('        solidity (sigma)  =   %6.3f \n',solidity)
fprintf('        Disk Loading      =   %6.2f lbs/ft^2\n',DL)
fprintf('         Figure of Merit  =   %6.2f \n',FM)
fprintf('              CT/sigma    =   %6.3f \n',CT_sig)
fprintf('              CQ/sigma    =   %6.4f \n',CQ_sig)
fprintf('              CH/sigma    =   %6.4f \n',CH_sig)
fprintf('Tip Mach number of adv blade=  %6.3f \n',Machtip)
fprintf('           Advance ratio  =   %6.3f \n',mu)
fprintf('Rotor thrust required (TPP)=   %6.0f lbs\n',T)
fprintf('      Rotor power required=   %6.0f h.p.\n',Protor)
fprintf('              Rotor torque=   %6.0f ft-lbs\n',Qrotor)
disp(' ')
disp(' ')
disp('press any key to continue...')
pause

clc
   flag=0;
else
   disp(' ')
   disp('         Enter a 1 or 2')
end
```

```
end

%   *** output to disk (text file) ***
disp(' ')
disp('         *** Saving data to disk ***')
pause(2)
diary on
diary off
delete diary
diary on

disp('         *** RESULTS ***')
eval(['disp("            ',filename1,'")'])
fprintf('      Forward Velocity    =   %6.0f kts\n',Vinf/1.69)
fprintf('          Temperature     =   %6.0f degs F\n',temp)
fprintf('       Pressure Altitude  =   %6.0f ft\n',PA)
fprintf('            Gross weight =   %6.0f lbs\n',GW)
fprintf('      Munber of blades    =   %6.0f \n',b)
fprintf('          Rotor Radius    =   %6.2f ft\n',R)
fprintf('            Blade Chord  =   %6.2f ft\n',cblade)
fprintf('            Blade twist  =   %6.2f degs\n',-1*twist*57.3)
fprintf('    Blade lift curve slope =   %6.2f \n',a)
fprintf('            Blade weight =   %6.2f lbs\n',wblade)
fprintf('      Rotational Velocity =   %6.2f rads/sec\n',omega)
fprintf('      Blade grip length   =   %6.2f ft\n',grip)
fprintf('          Hinge offset    =   %6.2f ft\n',e)
fprintf(' Equivalent flat plate area=   %6.2f ft^2\n',Afh)
fprintf('   Vertical projected area =   %6.2f ft^2\n',Afv)
fprintf('             Wing Area   =   %6.2f ft^2\n',Swing)
fprintf('             Wing Span   =   %6.2f ft\n',bwing)
fprintf('             Wing CL     =   %6.2f \n',CLwing)
fprintf('             Wing CDo    =   %6.4f \n',CDowing)
fprintf('   Wing Efficicency Factor =   %6.2f \n',ewing)
fprintf('   Horizontal tail area    =   %6.2f ft^2\n',Shoriz)
fprintf('   Horizontal tail span    =   %6.2f ft\n',bhoriz)
fprintf('    Horizontal tail CL     =   %6.2f \n',CLhoriz)
fprintf('    Horizontal tail CDo    =   %6.4f \n',CDohoriz)
fprintf('      Vertical tail area  =   %6.2f ft^2\n',Svert)
fprintf('      Vertical tail span  =   %6.2f ft\n',bvert)
fprintf('      Vertical tail CL    =   %6.2f \n',CLvert)
fprintf('      Vertical tail CDo   =   %6.4f \n',CDovert)
fprintf('          Fuselage drag   =   %6.0f lbs\n',Dfuse)
fprintf('             Rotor drag  =   %6.0f lbs\n',Drotor)
fprintf('             Wing lift   =   %6.0f lbs\n',Lwing)
fprintf('             Wing drag   =   %6.0f lbs\n',Dwing)
fprintf('   Horizontal tail lift    =   %6.0f lbs\n',Lhoriz)
fprintf('   Horizontal tail drag    =   %6.0f lbs\n',Dhoriz)
fprintf('   Vertical tail side force=   %6.0f lbs\n',Lvert)
fprintf('      Vertical tail drag  =   %6.0f lbs\n',Dvert)
fprintf('         Auxiliary thrust =   %6.0f lbs\n',Taux)
fprintf(' Vertical Auxiliary thrust =   %6.0f lbs\n',vertaux)
fprintf('         Tip path angle   =   %6.2f degs\n',alphaT*57.3)
fprintf('      Rotor coning angle  =   %6.2f degs\n',betao*57.3)
fprintf('Location of mean thrust(r/R)=  %6.2f \n',rT2)
fprintf(' Collective pitch at .7 r/R=   %6.2f degs\n',thetao*57.3)
fprintf('1st lat cyclic term-A1(deg)=   %6.2f \n',theta1c*57.3)
fprintf('1st long cyclic term-B1(deg)=  %6.2f \n',theta1s*57.3)
fprintf('2/rev cyclic term (deg)    =   %6.2f \n',theta2c*57.3)
fprintf('3/rev cyclic term (deg)    =   %6.2f \n',theta3s*57.3)
fprintf('3/rev offset (deg)         =   %6.2f \n',theta3Poffset*57.3)
fprintf('                solidity  =   %6.3f \n',solidity)
fprintf('             Disk loading =   %6.2f lbs/ft^2\n',DL)
fprintf('          Figure of Merit =   %6.2f \n',FM)
fprintf('              CT/sigma    =   %6.3f \n',CT_sig)
fprintf('              CQ/sigma    =   %6.4f \n',CQ_sig)
fprintf('              CH/sigma    =   %6.4f \n',CH_sig)
fprintf('Tip Mach of the adv. blade =   %6.3f \n',Machtip)
fprintf('          Advance Ratio   =   %6.3f \n',mu)
fprintf('Rotor thrust required(TPP) =   %6.0f lbs\n',T)
fprintf('   Rotor power required    =   %6.0f h.p.\n',Protor)
fprintf('          Rotor Torque    =   %6.0f ft-lbs\n',Qrotor)
diary off

%        *** Output to disk (.mat file containing matrix variables:
%        r, psi, vi, theta, thetaXP, betat, alpha, Tpsi, Mpsi, DMpsi, dT, dM
%        dD, CLplot, CDplot) ***

%        *** Configuring variables for output ***

theta=theta*57.3;
thetaXP=thetaXP*57.3;
betat=[betat twist*(0.7-(Reff+(R-Reff)/2)/R)]*57.3;
alpha=alpham*57.3;,alpha=[alpha zeros(size(psi))];
Mpsi=Mpsi(:,length(Mpsi(1,:))-1);
dM=[dM ddM];
psi=psi*57.3;
r=[r (R-(R-Reff)/2)];
vi=[vi 0];
dr=dr*12;

clc
flag1=1;
while flag1 > 0
disp(' ')
wantplots=input('Do you want to view plots, 1.  yes or 2.  no?');
   if wantplots==2,
      flag1=0;
   elseif wantplots==1,
      plotter
      flag1=0;
   else
   disp(' ')
   disp('        Enter a 1 or 2')
   end
end

clc
disp(' ')
disp('        *** PERFORMANCE OUTPUT DATA INSTRUCTIONS ***')
disp(' ')
disp('     A. Single value data saved to a file named:')
eval(['disp(''         ''',filename1,'.prf'''')'])
```

```
disp(' ')
disp('      B. This is a text file, use the tyoe command to view the')
disp('      file or use a text editor to view/print the file.')
eval(['flag=exist(''',filename1,'.prf'');'])
    if flag < 1,
        eval(['!rename diary ',filename1,'.prf'''])
    else
        eval(['!del ',filename1,'.prf'''])
        eval(['!rename diary ',filename1,'.prf'''])
    end

filename2=[filename1 '_p'];
disp(' ')
disp('      C. Matrix and vector data saved to a file named:')
eval(['disp(''          ''',filename2,'.mat'''')'])
disp(' ')
disp('      D. This is a ".mat" binary file, use the load command')
disp('      to retrieve the data for plotting.')
eval(['save ',filename2,' r psi vi theta thetaXP betat alpha Tpsi Mpsi DMpsi dT dM dD CLplot
CDplot'])
disp(' ')
disp('         *** END OF PERFORMANCE ***')
disp(' ')
disp(' ')
disp('         press any key to continue...')
pause
```

APPENDIX D. PLOTTER.M

```
%    *** Plotting Routine added 28 July 2003 some code copied/modified from lines 28-72 of
create_plots
%      JANRAD v6.0 ***
clc
disp(' ')
disp('      **** Available Plots ****')
disp(' ')
disp('      1.  Rotor Azimuth Angle v. Cyclic Pitch')
disp('      2.  Blade Azimuth Position v. Airload')
disp('      3.  Rotor Airload Distribution')
disp('      4.  Blade Azimuth Position v. Lift')
disp('      5.  Rotor Lift Distribution')
disp('      6.  Angle of Attack Distribution')
disp('      7.  Rotor Tip Path Plane Stall pattern and Thrust Distribution')
disp('      8.  Rotor Trim Lissajous Figure')
disp('      9.  Lift Distribution (Contour Plot)')
disp('      10. CL Distribution (Contour Plot)')
disp('      11. Mach Number Distribution (Contour Plot)')
disp(' ')
disp(' ')
flag2=1;
drinches=dr;
while flag2 > 0
   plotchoice=input('Please choose a plot by entering 1-11.');

   if plotchoice==1,
      flag2=0;
      if twoPinput==0 &  threePinput==0
         figure(1)
         plot(psi,thetaXP)
         axis([0,360,-10,30]);
         legend('1P Harmonic Cyclic Pitch')
         title(['Rotor Azimuth Angle v. Harmonic Cyclic Pitch for ',num2str(Vinf/1.68781),...
               ' Kts, advance ratio=',num2str(mu)])
         xlabel('Rotor Azimuth Angle (degs)'),ylabel('Cyclic Pitch (degs)')
      elseif rvrCP==1
         figure(1)
         thetao=thetao.*ones(size(theta2Pr));
         plot(psi,thetao*57.3,'-',psi,(-theta-thetao*57.3),':',psi,theta2Pr*57.3,'--',psi,thetaXP,'*')
         axis([0,360,-25,20]);
         legend('Collective Pitch','1P Cyclic Pitch','2P Cyclic Pitch','Coll + 1P + 2P signal')
         title(['Rotor Azimuth Angle v. RVR Harmonic Cyclic Pitch for ',num2str(Vinf/1.68781),...
               ' Kts, advance ratio=',num2str(mu),', 2P input=',num2str(twoPinput),...
               ' degs'])
         xlabel('Rotor Azimuth Angle (degs)'),ylabel('Cyclic Pitch (degs)')
      else
         figure(1)
         plot(psi,theta,psi,theta2Pr*57.3,'--',psi,theta3Pr*57.3,':',psi,thetaXP,'*')
         axis([0,360,-15,35]);
         legend('1P Cyclic Pitch','2P Cyclic Pitch',...
            '3P Cyclic Pitch','1P + 2P + 3P signal')
         title(['Rotor Azimuth Angle v. Harmonic Cyclic Pitch for ',num2str(Vinf/1.68781),...
                  ' Kts, advance ratio=',num2str(mu),', 2P input=',num2str(twoPinput),...
                  ' degs, 3P input=',num2str(threePinput),...
                  ' degs, 3P offset=',num2str(threePoffset),' degs'])
              xlabel('Rotor Azimuth Angle (degs)'),ylabel('Cyclic Pitch (degs)')
            end

          elseif plotchoice==2,
            flag2=0;
          figure(2)
            subplot(3,3,8)
            plot(r./R,dT(1,:)./drinches,'k'),grid
            title('Psi = 0 deg ')
            xlabel('Blade Position r/R');ylabel('Airload (Lb/in)');

            subplot(3,3,2)
            plot(r./R,dT(floor(naz/2),:)./drinches,'k'),grid
            title('Psi = 180 deg ')
            xlabel('Blade Position r/R'); ylabel('Airload (Lb/in)');

            subplot(3,3,6)
            plot(r./R,dT(floor(naz/4),:)./drinches,'k'),grid
            title('Psi = 90 deg ')
            xlabel('Blade Position r/R');ylabel('Airload (Lb/in)');

            subplot(3,3,4)
            plot(r./R,dT(floor(3*naz/4),:)./drinches,'k'),grid
            title('Psi = 270 deg ')
            xlabel('Blade Position r/R'); ylabel('Airload (Lb/in)');
            gtext(['Blade Position v. Airload for ',num2str(Vinf/1.68781),' Kts, advance
ratio=',num2str(mu),...
              ', 2P input=',num2str(twoPinput),' degs, 3P input=',...
              num2str(threePinput), ' degs, 3P offset=',num2str(threePoffset),' degs'])

          elseif plotchoice==3,
            flag2=0;
          figure(3)
          [th,r1] = meshgrid((-180:360/naz:180)*pi/180,r/R);
          [x,y] = pol2cart(th,r1);
          dT1=[dT; dT(1,:)];
          for i=1:naz+1
            dT1(i,:)=dT1(i,:)./(drinches);
          end
          mesh(x',y',dT1)
          view(315,60)
          xlabel('Starboard'); ylabel('Aft'); zlabel('Aero Load, Lb/in');
          title(['Airload Distribution At ',num2str(Vinf/1.68781),' Kts, advance ratio=',num2str(mu),...
              ', 2P input=',num2str(twoPinput),' degs, 3P input=',...
              num2str(threePinput), ' degs, 3P offset=',num2str(threePoffset),' degs'])

          elseif plotchoice==4,
            flag2=0;
          figure(4)
           subplot(3,3,8)
           plot(r./R,dN(1,:)),grid
           title('Psi = 0 deg ')
           xlabel('r/R'); ylabel('Lift,Lb');
```

```
  subplot(3,3,2)
  plot(r./R,dN(floor(naz/2),:)),grid
  title('Psi = 180 deg ')
  xlabel('r/R'); ylabel('Lift,Lb');

  subplot(3,3,6)
  plot(r./R,dN(floor(naz/4),:)),grid
  title('Psi = 90 deg ')
  xlabel('r/R'); ylabel('Lift,Lb');

  subplot(3,3,4)
  plot(r./R,dN(floor(3*naz/4),:)),grid
  title('Psi = 270 deg ')
  xlabel('r/R'); ylabel('Lift,lb');
  gtext(['Blade Position v. Lift for',num2str(Vinf/1.68781),' Kts, advance ratio=',num2str(mu),...
       ', 2P input=',num2str(twoPinput),' degs, 3P input=',...
       num2str(threePinput), ' degs, 3P offset=',num2str(threePoffset),' degs'])

elseif plotchoice==5,
   flag2=0;
figure(5)
[th,r1] = meshgrid((-180:360/naz:180)*pi/180,r/R);
[x,y] = pol2cart(th,r1);
dN1=[dN; dN(1,:)];
for i=1:naz+1
   dN1(i,:)=dN1(i,:);
end
mesh(x',y',dN1)
view(315,60)
xlabel('Starboard'); ylabel('Aft'); zlabel('Lift, Lb');
title(['Lift Distribution At ',num2str(Vinf/1.68781),' Kts, advance ratio=',num2str(mu),...
       ', 2P input=',num2str(twoPinput),' degs, 3P input=',...
       num2str(threePinput), ' degs, 3P offset=',num2str(threePoffset),' degs'])

elseif plotchoice==6,
   flag2=0;
   figure(6)
   alpha=alpham*57.3;
   alpha=alpha';
   dr1=(R-e)/nbe;
   r1=e:dr1:R-dr1;
   r1=r1+dr1/2;
   [th,r2]=meshgrid((0:360/naz:360)*pi/180,r1/R);
   [x,y]=pol2cart(th,r2);
   alpha=[alpha,alpha(:,1)];
   % The vector v may be altered to show user defined contour lines
   if rvrCP==1
      v=[-196,-194,-190,-180,-170,-18,-14,-10,-8,-4,-2,0,2,4,8,10,14,18];
   else
      v=[18,-16,-14,-12,-10,-8,-6,-4,-2,0,2,4,6,8,10,12,14,16];
   end
   h1=polar1([0,2*pi], [0,1]);
   delete(h1)
   hold on
```

```
                % Either [C,h2] equation may be used.  The second equation, shown in line 162, depicts the
hub area
                % [C,h2]=contour(x,y,alpha,v);

[C,h2]=contour(x(round(grip*12/dr):nbe,1:nbe+1),y(round(grip*12/dr):nbe,1:nbe+1),alpha(round(grip*12/
dr):nbe,1:nbe+1),v);
                clabel(C,h2);
                title(['Angle of Attack Distribution At ',num2str(Vinf/1.68781),' Kts, advance
ratio=',num2str(mu),...
                   ', 2P input=',num2str(twoPinput),' degs, TPP angle=',...
                   num2str(alphaT*57.3),' degs, 3P input=',num2str(threePinput), ' degs, 3P offset=',...
                   num2str(threePoffset),' degs'])
                hold on
                thetahub=[0:pi/270:2*pi];
                hub=(grip/R)./(1+0*cos(thetahub));
                   for i=1:.1:20
                      polar(thetahub,hub/i);
                      hold on
                   end
                hold off
                view(90,90)

             elseif plotchoice==7,
                flag2=0;
             figure(7)
                subplot(2,1,1)
                plot(psi,alpha(naz,1:naz)),set(gca,'XTick',[0:90:360],'YTick',[-8:2:14])
                xlabel('Azimuth Angle, degs'),ylabel('Tip Angle of Attack, degs')
                title(['Rotor Tip Path Plane Stall Pattern, advance ratio=',num2str(mu)])
                text(120,10,'Ideal')
                hold on
                psiplot1=   [0,45,45,135,135,180,210,250,270,290,360];
                ideal=      [10,7,-.5,-.5,7,10,11,12,12.5,13,10.5];
                plot(psiplot1,ideal,'r--')
                hold off


                dTplot=sum(dT')/(rho*Adisk*(omega*R)^2);
                subplot(2,1,2)
                plot(psi,dTplot),set(gca,'XTick',[0:90:360])
                xlabel('Azimuth Angle, degs'),ylabel('Thrust Coefficient (one blade)')
                title(['Actual and Ideal Thrust Distribution at Stall, advance ratio=',num2str(mu)])
                text(180,.003,'Ideal')
                hold on
                psiplot2=   [0      45   45   135    135   180   210   240   270    315    340    360];
                ideal2=     [.003 .0048 .0001 .0001  .0048  .003  .002  .001  .0009  .0015  .002   .0029];
                plot(psiplot2,ideal2,'r--')
                hold off
                gtext([num2str(Vinf/1.68781),' Kts, advance ratio=',num2str(mu),...
                   ', 2P input=',num2str(twoPinput),' degs, TPP angle=',...
                   num2str(alphaT),' degs, 3P input=',num2str(threePinput), ' degs, 3P offset=',...
                   num2str(threePoffset),' degs'])

             elseif plotchoice==8,
                flag2=0;
             figure(8)
```

```
            Mpsi1=[Mpsi; Mpsi(1,1)];
            % if desired convert to in * lbs
            Mpsi1=Mpsi1*12;
            theta1=[theta; theta(1,1)];
            plot(Mpsi1,theta1)
            axis([-1e6,3e6,0,35]);
            theta1=theta1*pi/180;
            alphaT=alphaT*57.3

            % determine Lissajous area
            area=trapz(Mpsi*12,theta/57.3);
            area=abs(area);

            xlabel('Thrust Moment, in lb'),ylabel('Blade Angle, degs')
            title(['Lissajou Figure ',num2str(Vinf/1.68781),' Kts, advance ratio=',num2str(mu),...
                 ', 2P input=',num2str(twoPinput),' degs, Lissajou Area=',num2str(area),' in lbs, TPP
angle=',num2str(alphaT),' degs, 3P input=',...
                 num2str(threePinput), ' degs, 3P offset=',num2str(threePoffset),' degs'])

            elseif plotchoice==9,
               flag2=0;
            figure(9)
            dr1=(R-e)/nbe;
            r1=e:dr1:R-dr1;
            r1=r1+dr1/2;
            [th,r2]=meshgrid((0:360/naz:360)*pi/180,r1/R);
            [x,y]=pol2cart(th,r2);


            for i=1:naz;
               for j=1:naz;
                  dT2(i,j)=dT(i,j)./(drinches);
               end
            end
            dT3=[dT2; dT2(1,:)];
            dT3=dT3';
            v=[-5,-4,-3,-2,-1,0,4,8,12,16,20,24,28,32,36,40,44,48,52,56,60,64,68];
            h1=polar1([0,2*pi], [0,1]);
            delete(h1)
            hold on
            % Either [C,h2] equation may be used.  The second depicts the hub area
            % [C,h2]=contour(x,y,dT3,v);

[C,h2]=contour(x(round(grip*12/dr):nbe,1:nbe+1),y(round(grip*12/dr):nbe,1:nbe+1),dT3(round(grip*12/dr
):nbe,1:nbe+1),v);
            clabel(C,h2);
            title(['Airload Distribution At ',num2str(Vinf/1.68781),' Kts, advance ratio=',num2str(mu),...
                 ', 2P input=',num2str(twoPinput),' degs, TPP angle=',...
                 num2str(alphaT*57.3),' degs, 3P input=',num2str(threePinput), ' degs, 3P offset=',...
                 num2str(threePoffset),' degs'])
            hold on
               thetahub=[0:pi/270:2*pi];
               hub=(grip/R)./(1+0*cos(thetahub));
               for i=1:.1:20
                  polar(thetahub,hub/i);
                  hold on
```

```
          end
        hold off
        view(90,90)

        elseif plotchoice==10,
          flag2=0;
        figure(10)
        dr1=(R-e)/nbe;
        r1=e:dr1:R-dr1;
        r1=r1+dr1/2;
        [th,r2]=meshgrid((0:360/naz:360)*pi/180,r1/R);
        [x2,y2]=pol2cart(th,r2);

        CLplot1=CLplot';
        CLplot1=[CLplot1,CLplot1(:,1)];
        % The vector v may be altered to show user defined contour lines
        % v=[-1,-.8,-.6,-.4,-.3,-.2,-.1,0,.1,.2,.3,.4,.6,.8,1.0,1.2,1.4];
        v=[-1,-.8,-.6,-.4,-.2,0,.2,.4,.6,.8,1.0,1.2,1.4];
        h1=polar1([0,2*pi], [0,1]);
        delete(h1)
        hold on

[C,h2]=contour(x2(round(grip*12/dr):nbe,1:nbe+1),y2(round(grip*12/dr):nbe,1:nbe+1),CLplot1(round(gri
p*12/dr):nbe,1:nbe+1),v);
        clabel(C,h2);
        title(['CL Distribution At ',num2str(Vinf/1.68781),' Kts, advance ratio=',num2str(mu),...
            ', 2P input=',num2str(twoPinput),' degs, TPP angle=',...
            num2str(alphaT*57.3),' degs, 3P input=',num2str(threePinput), ' degs, 3P offset=',...
            num2str(threePoffset),' degs'])
        hold on
          thetahub=[0:pi/270:2*pi];
          hub=(grip/R)./(1+0*cos(thetahub));
          for i=1:.1:20
            polar(thetahub,hub/i);
            hold on
          end
        hold off
        view(90,90)

        elseif plotchoice==11,
          flag2=0;
        figure(11)
        dr1=(R-e)/nbe;
        r1=e:dr1:R-dr1;
        r1=r1+dr1/2;
        [th,r2]=meshgrid((0:360/naz:360)*pi/180,r1/R);
        [x3,y3]=pol2cart(th,r2);

        machplot1=machplot';
        machplot2=[machplot1,machplot1(:,1)];

        v=[-.4,-.3,-.2,-.1,0,.1,.2,.3,.4,.5,.6,.7,.8,.9,1];
        h1=polar1([0,2*pi], [0,1]);
        delete(h1)
        hold on
        [C,h2]=contour(x3,y3,machplot2,v);
```

```
          clabel(C,h2);
          title(['Mach     Number     Distribution     At     ',num2str(Vinf/1.68781),'     Kts,     advance
ratio=',num2str(mu)])
          hold on
             thetahub=[0:pi/270:2*pi];
             hub=(grip/R)./(1+0*cos(thetahub));
             for i=1:.1:20
                polar(thetahub,hub/i);
                hold on
             end
          hold off
          view(90,90)

          else
                   disp(' ')
                   disp(' *** Enter a 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, or 11 *** ')
          end
       end

       clc
       flag3=1;
       while flag3 > 0
       disp(' ')
       wantmoreplots=input('Do you want to view additional plots, 1.  yes or 2.  no?');
          if wantmoreplots==2,
             flag3=0;
          elseif wantmoreplots==1
             plotter
             flag3=0;
          else
             disp(' ')
             disp('         Enter a 1 or 2')
          end
       end

       %   *** End of Plotting Routine ***
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E. POLAR1.M

```
function hpol = polar(theta,rho,line_style)
%POLAR  Polar coordinate plot.
%   POLAR(THETA, RHO) makes a plot using polar coordinates of
%   the angle THETA, in radians, versus the radius RHO.
%   POLAR(THETA,RHO,S) uses the linestyle specified in string S.
%   See PLOT for a description of legal linestyles.
%
%   See also PLOT, LOGLOG, SEMILOGX, SEMILOGY.

%   Copyright 1984-2002 The MathWorks, Inc.
%   $Revision: 5.22 $  $Date: 2002/04/08 21:44:28 $

if nargin < 1
   error('Requires 2 or 3 input arguments.')
elseif nargin == 2
   if isstr(rho)
      line_style = rho;
      rho = theta;
      [mr,nr] = size(rho);
      if mr == 1
         theta = 1:nr;
      else
         th = (1:mr)';
         theta = th(:,ones(1,nr));
      end
   else
      line_style = 'auto';
   end
elseif nargin == 1
   line_style = 'auto';
   rho = theta;
   [mr,nr] = size(rho);
   if mr == 1
      theta = 1:nr;
   else
      th = (1:mr)';
      theta = th(:,ones(1,nr));
   end
end
if isstr(theta) | isstr(rho)
   error('Input arguments must be numeric.');
end
if ~isequal(size(theta),size(rho))
   error('THETA and RHO must be the same size.');
end

% get hold state
cax = newplot;
next = lower(get(cax,'NextPlot'));
hold_state = ishold;

% get x-axis text color so grid is in same color
```

```
tc = get(cax,'xcolor');
ls = get(cax,'gridlinestyle');

% Hold on to current Text defaults, reset them to the
% Axes' font attributes so tick marks use them.
fAngle  = get(cax, 'DefaultTextFontAngle');
fName   = get(cax, 'DefaultTextFontName');
fSize   = get(cax, 'DefaultTextFontSize');
fWeight = get(cax, 'DefaultTextFontWeight');
fUnits  = get(cax, 'DefaultTextUnits');
set(cax, 'DefaultTextFontAngle',  get(cax, 'FontAngle'), ...
    'DefaultTextFontName',   get(cax, 'FontName'), ...
    'DefaultTextFontSize',   get(cax, 'FontSize'), ...
    'DefaultTextFontWeight', get(cax, 'FontWeight'), ...
    'DefaultTextUnits','data')

% only do grids if hold is off
if ~hold_state

% make a radial grid
    hold on;
    maxrho = max(abs(rho(:)));
    hhh=plot([-maxrho -maxrho maxrho maxrho],[-maxrho maxrho maxrho -maxrho]);
    set(gca,'dataaspectratio',[1 1 1],'plotboxaspectratiomode','auto')
    v = [get(cax,'xlim') get(cax,'ylim')];
    ticks = sum(get(cax,'ytick')>=0);
    delete(hhh);
% check radial limits and ticks
    rmin = 0; rmax = v(4); rticks = max(ticks-1,2);
    if rticks > 5   % see if we can reduce the number
        if rem(rticks,2) == 0
            rticks = rticks/2;
        elseif rem(rticks,3) == 0
            rticks = rticks/3;
        end
    end

% define a circle
    th = 0:pi/50:2*pi;
    xunit = cos(th);
    yunit = sin(th);
% now really force points on x/y axes to lie on them exactly
    inds = 1:(length(th)-1)/4:length(th);
    xunit(inds(2:2:4)) = zeros(2,1);
    yunit(inds(1:2:5)) = zeros(3,1);
% plot background if necessary
    if ~isstr(get(cax,'color')),
       patch('xdata',xunit*rmax,'ydata',yunit*rmax, ...
             'edgecolor',tc,'facecolor',get(gca,'color'),...
             'handlevisibility','off');
    end

% draw radial circles
          c82 = cos(82*pi/180);
          s82 = sin(82*pi/180);
          rticks=1;
    rinc = (rmax-rmin)/rticks;
             for i=(rmin+rinc):rinc:rmax
       hhh = plot(xunit*i,yunit*i,ls,'color',tc,'linewidth',1,...
           'handlevisibility','off');
       text((i+rinc/20)*c82,(i+rinc/20)*s82, ...
           ['  ' num2str(i)],'verticalalignment','bottom',...
           'handlevisibility','off')
             end
             set(hhh,'linestyle','-') % Make outer circle solid

% plot spokes
    th = (1:6)*2*pi/12;
    cst = cos(th); snt = sin(th);
    cs = [-cst; cst];
    sn = [-snt; snt];
    plot(rmax*cs,rmax*sn,ls,'color',tc,'linewidth',1,...
         'handlevisibility','off')

% annotate spokes in degrees
    rt = 1.1*rmax;
    for i = 1:length(th)
       text(rt*cst(i),rt*snt(i),int2str(i*30),...
            'horizontalalignment','center',...
            'handlevisibility','off');
       if i == length(th)
           loc = int2str(0);
       else
           loc = int2str(180+i*30);
       end
       text(-rt*cst(i),-rt*snt(i),loc,'horizontalalignment','center',...
            'handlevisibility','off')
    end

% set view to 2-D
    view(2);
% set axis limits
    axis(rmax*[-1 1 -1.15 1.15]);
end

% Reset defaults.
set(cax, 'DefaultTextFontAngle', fAngle , ...
    'DefaultTextFontName',   fName , ...
    'DefaultTextFontSize',   fSize, ...
    'DefaultTextFontWeight', fWeight, ...
    'DefaultTextUnits',fUnits );

% transform data to Cartesian coordinates.
xx = rho.*cos(theta);
yy = rho.*sin(theta);

% plot data on top of grid
if strcmp(line_style,'auto')
    q = plot(xx,yy);
else
    q = plot(xx,yy,line_style);
end
```

```
if nargout > 0
    hpol = q;
end
if ~hold_state
    set(gca,'dataaspectratio',[1 1 1]), axis off; set(cax,'NextPlot',next);
end
set(get(gca,'xlabel'),'visible','on')
set(get(gca,'ylabel'),'visible','on')
```

APPENDIX F. THRCALC.M

```
% thrcalc calculates the total thrust along a blade at
% each azimuth (psi) location
% disp('begin thrcalc')

Up=zeros(size(psi*r));
Ut=zeros(size(Up));
dT=zeros(size(Up));
ddT=zeros(size(psi));
dN=zeros(size(Up));
ddN=zeros(size(psi));

for i=1:length(psi),
Up(i,:)=vi.*cos(betao)+Vinf*sin(alphaT)*cos(betao)+Vinf*cos(alphaT)*sin(betao)*cos(psi(i));
   Ut(i,:)=r.*omega+Vinf*cos(alphaT)*sin(psi(i));
   phi=atan2(Up(i,:),Ut(i,:));
   mach = (Vtip.*cos(alphaT).*r./R+Vinf.*sin(psi(i)))/(49.05*sqrt(temp+460));
   %   alpha=theta(i)+betat-phi;
   alpha=thetaXP(i)+betat-phi;

      if airfoil==1,
         [CL,CD]=hh02clcd(alpha);
      % additional airfoils added for experimentation
      elseif airfoil==2,
         [CL,CD]=vr12clcd(alpha);
      elseif airfoil==3,
         [CL,CD]=sc1095r8clcd(alpha,mach);
      elseif airfoil==4,
         [CL,CD]=oo12clcd(alpha,mach);
      elseif airfoil==5,
         [CL,CD]=rvrclcd(alpha,mach);
      end

   dT(i,:)=0.5*rho.*cblade.*dr.*(Up(i,:).^2+Ut(i,:).^2).*(CL.*cos(phi)-CD.*sin(phi));
   Tpsi(i)=sum(dT(i,:));
   dN(i,:)=0.5*rho.*cblade.*dr.*(Up(i,:).^2+Ut(i,:).^2).*(CL.*cos(alpha)+CD.*sin(alpha));
   Npsi(i)=sum(dT(i,:));

% Populate CLplot and CDplot matrix
   CLplot(i,:)=CL;
   CDplot(i,:)=CD;
% Populate Mach matrix
   machplot(i,:)=mach;
% Populate alpha matrix
   alpham(i,:)=alpha;

%   *** calculations for tip loss area ***

   Uptip=Vinf*sin(alphaT)*cos(betao)+Vinf*cos(alphaT)*sin(betao)*cos(psi(i));
   Uttip=(R-(R-Reff)/2)*omega+Vinf*cos(alphaT)*sin(psi(i));
   phitip=atan2(Uptip,Uttip);
   alphatip=theta(i)+betat-phitip;
   %ddT(i)=0.5*rho*cblade*(R-Reff)*(Uptip^2)*(-.009*sin(phitip));
```

```
    ddT(i)=0.5*rho*cblade*(0.5+0.5*cos(2*psi(i)))*(R-Reff)*(Uptip^2+Uttip^2)*(-
.009*sin(phitip));
    Tpsi(i)=Tpsi(i)+ddT(i);
    ddN(i)=0.5*rho*cblade*(0.5+0.5*cos(2*psi(i)))*(R-Reff)*(Uptip^2+Uttip^2)*(-
.009*sin(alphatip(i)));
    Npsi(i)=Npsi(i)+ddN(i);
end

machplot2=machplot';
CLplot2=CLplot';
```

APPENDIX G. TMCALC.M

```
% tmcalc calculates the total thrust moment along a blade
% at each azimuth (psi) location
% disp('begin tmcalc')

Up=zeros(size(psi*r));
Ut=zeros(size(Up));
dM=zeros(size(Up));
ddM=zeros(size(psi));

for i=1:length(psi),
   Up(i,:)=vi.*cos(betao)+Vinf*sin(alphaT)*cos(betao)+Vinf*cos(alphaT)*sin(betao)*cos(psi(i));
      Ut(i,:)=r.*omega+Vinf*cos(alphaT)*sin(psi(i));
      phi=atan2(Up(i,:),Ut(i,:));
%       alpha=theta(i,k)+betat-phi;
      alpha=thetaXP(i,k)+betat-phi;
      if airfoil==1,
         [CL,CD]=hh02clcd(alpha);
      % additional airfoils added for experimentation
      elseif airfoil==2,
         [CL,CD]=vr12clcd(alpha);
      elseif airfoil==3,
         [CL,CD]=sc1095r8clcd(alpha,mach);
      elseif airfoil==4,
         [CL,CD]=oo12clcd(alpha,mach);
      elseif airfoil==5,
         [CL,CD]=rvrclcd(alpha,mach);
      end

% Populate CLplot and CDplot matrix
   CLplot(i,:)=CL;
   CDplot(i,:)=CD;
% Populate alpham matrix
   alpham(i,:)=alpha;


   dM(i,:)=0.5*rho*cblade.*r*dr.*(Up(i,:).^2+Ut(i,:).^2).*(CL.*cos(phi)-CD.*sin(phi));
   Mpsi(i,k)=sum(dM(i,:));

%  *** calculations for tip loss areas ***

   Uptip=Vinf*sin(alphaT)*cos(betao)+Vinf*cos(alphaT)*sin(betao)*cos(psi(i));
   Uttip=(R-(R-Reff)/2)*omega+Vinf*cos(alphaT)*sin(psi(i));
   phitip=atan2(Uptip,Uttip);

   ddM(i)=0.5*rho*cblade*(R-(R-Reff)/2)*(R-Reff)*(Uptip^2+Uttip^2)*(-.009*sin(phitip));
   Mpsi(i,k)=Mpsi(i,k)+ddM(i);
end

CLplot2=CLplot';
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX H. TRIM.M

```
%   Rotor trim subroutine

disp(' ')
disp('          *** EXECUTING ROTOR TRIM ROUTINE ***')


%   *** calculation of required parameters ***

rho=.002377*(-.000031*PA+(-.002*temp+1.118));

% Hardwire rho for high and hot conditions
if PA==4000 & temp==95
   rho=.00192
else
end

%   *** first guess at rotor profile drag ( H force ) ***

if Vinf < 16.9,
   Drotor=0;
else
   Drotor=Vinf*(rho/.002377);
end

q=0.5*rho*Vinf^2;
Adisk=pi*R^2;
Vtip=omega*R;
Dfuse=q*Afh;
CDwing=CDowing+(CLwing^2/(ewing*pi*(bwing^2/Swing)));
CDhoriz=CDohoriz+(CLhoriz^2/(.8*pi*(bhoriz^2/Shoriz)));
CDvert=CDovert+(CLvert^2/(.8*pi*(bvert^2/Svert)));
Dwing=q*CDwing*Swing;
Dhoriz=q*CDhoriz*Shoriz;
Dvert=q*CDvert*Svert;
Lwing=q*CLwing*Swing;
Lhoriz=q*CLhoriz*Shoriz;
Lvert=q*CLvert*Svert;
Lftotal=Lwing+Lhoriz;
Dftotal=Dfuse+Dwing+Dhoriz+Dvert;

%   *** thrust calculation ***

% Check percentage of wing lift versus rotor thrust

if helochoice==2 & Vinf > 16.9
   clc
   LoverT=Lftotal/GW;
   disp('The wing is carrying '),disp(LoverT*100),disp('percent of the aircraft weight.')
   flag=1;
         while flag > 0
            disp('')
            disp('Do you want to change the percentage of aircraft weight carried by the wing? ')
```

```
                    disp('1. continue with current wing settings or 2. change percentage of aircraft weight
carried by the wing by varying wing area')
                    wingchoice=input('Enter a 1 or 2...');
                    while isempty(wingchoice),
                      disp(' ')
                      disp('You must enter a 1 or 2')
                      disp('Do you want to change the percentage of aircraft weight carried by the wing? ')
                      disp('1. continue with current wing settings or 2. change percentage of aircraft weight
carried by the wing by varying wing area')
                      wingchoice=input('Enter a 1 or 2...');
                    end
                    if wingchoice~=1 & wingchoice~=2
                      disp(' ')
                      disp(' *** Enter a 1 or 2 ***')
                    elseif wingchoice==2
                      wpercent=input('Enter desired percentage of aircraft weight carried by wing: ');
                      lifterror=wpercent/100*GW-Lftotal;
                      if lifterror > 0
                        while lifterror > .05*wpercent/100*GW
                          Swing=Swing + 5;
                          Lwing=q*CLwing*Swing;
                          Lftotal=Lwing+Lhoriz;
                          lifterror=wpercent/100*GW-Lftotal;
                        end
                        lifterror=0;
                        disp('New wing area is '),disp(Swing),disp(' ft^2')
                        disp('Press any key to continue ...')
                        pause
                      elseif lifterror < 0
                        while lifterror < .05*wpercent/100*GW
                          Swing=Swing - 5;
                          Lwing=q*CLwing*Swing;
                          Lftotal=Lwing+Lhoriz;
                          lifterror=wpercent/100*GW-Lftotal;
                        end
                        lifterror=0;
                        disp('New wing area is '),disp(Swing),disp(' ft^2')
                        disp('Press any key to continue ...')
                        pause
                      end
                      flag=0;
                    elseif wingchoice==1
                      disp('Wing settings unchanged')
                      disp('Press any key to continue ...')
                      pause
                      flag=0;
                    end
                  end

        end

        %   if required, define alphaT fixed quantity
        if alphaTfix==1
          alphaT=alphaTinput;
        else
          alphaT=atan2((Dftotal+Drotor), (GW-Lftotal));
```

```
    end

    % Determine thrust parameters for low speed flight
    if Vinf < 16.9,
       % Determine augmented vertical thrust capability
       clc
       disp('JANRAD is about to perform low speed performance analysis.')
       disp('')
       flag=1;
       while flag > 0
          disp('')
          disp('Are any auxiliary vertical thrust devices installed on the aircraft,')
          disp('i.e., Lift Fans or Direct Jet thrusters?')
          disp('')
          vertauxchoice=input('1. yes or 2. no');
          while isempty(vertauxchoice),
             disp(' ')
             disp('You must enter a 1 or 2')
             disp('Are there any auxiliary vertical thrust devices installed on the aircraft,')
             disp('i.e., Lift Fans or Direct Jet thrusters?')
             disp('')
             vertauxchoice=input('1. yes or 2. no');
          end
          if vertauxchoice~=1 & vertauxchoice~=2
             disp(' ')
             disp(' *** Enter a 1 or 2 ***')
          elseif vertauxchoice==1
             vertaux=input('Enter amount of thrust in pounds provided by the vertical auxiliary thrust
devices: ');
             GW=GW-vertaux;
             disp('Input parameters adjusted to reflect auxiliary vertical thrust.')
             flag=0;
          elseif vertauxchoice==2
             vertaux=0;
             disp('No vertical thrust devices installed')
             disp('Press any key to continue ...')
             pause
             flag=0;
          end
       end

       T=(1+(0.3*Afv/Adisk))*GW;
       CT=T/(Adisk*rho*Vtip^2);
    % Determine thrust parmeters for nominal flight speeds
    else
       T=(GW-Lftotal)/cos(alphaT);
       CT=T/(Adisk*rho*Vtip^2);
       vertaux=0;
    end

    % Added for Compound/RVR auxillary thrust analysis
    if helochoice==2 & Vinf > 16.9
       clc
       if Tauxchoice==1
          Taux=(Dfuse+Dwing+Dhoriz+Dvert);
          %   if required, define alphaT fixed quantity
```

```
            if alphaTfix==1
               alphaT=alphaTinput;
            else
               alphaT=atan2((Dftotal+Drotor), (GW-Lftotal));
            end
         else
            if Taux<((Dfuse+Dwing+Dhoriz+Dvert)-sin(alphaT)*T)
               disp('***   Auxillary thrust will not overcome drag at this airspeed,   ***')
               disp('      Total drag = '),disp(Dfuse+Dwing+Dhoriz+Dvert)
               disp('      Thrust deficit ='),disp(Dfuse+Dwing+Dhoriz+Dvert-sin(alphaT)*T-Taux)
               flag=1;
               while flag > 0
                  disp('')
                  disp('Do you want to change the auxillary thrust value? ')
                  disp('1. continue with current auxillary thrust value or 2. set auxillary thrust to equal
drag')
                  Tauxchoice=input('Enter a 1 or 2...');
                  while isempty(Tauxchoice),
                     disp(' ')
                     disp('You must enter a 1 or 2')
                     disp('')
                     disp('Do you want to change the auxillary thrust value? ')
                     disp('')
                     disp('1. continue with current auxillary thrust value or 2. set auxillary thrust to equal
drag')
                     Tauxchoice=input('Enter a 1 or 2...');
                  end
                  if Tauxchoice~=1 & Tauxchoice~=2
                     disp(' ')
                     disp(' *** Enter a 1 or 2 ***')
                  elseif Tauxchoice==2
                     Taux=(Dfuse+Dwing+Dhoriz+Dvert);
                     flag=0;
                     disp('Auxillary thrust change to equal total drag')
                     disp('')
                     disp('Press any key to continue ...')
                     pause
                  elseif Tauxchoice==1
                     disp('Auxillary thrust deficit is '),disp(Dfuse+Dwing+Dhoriz+Dvert-sin(alphaT)*T-
Taux),disp(' lbs.')
                     disp('***      Results may be erroneous.      ***')
                     disp('')
                     disp('Press any key to continue ...')
                     pause
                     flag=0;
                  end
               end
            end
            % define alphaT fixed quantity
            if alphaTfix==1
               alphaT=alphaTinput;
            else
               alphaT=atan2((Dftotal+Drotor), (GW-Lftotal));
            end
         end
      end
```

```
        Dftotal=(Dfuse+Dwing+Dhoriz+Dvert)-Taux;
        mu=Vinf*cos(alphaT)/Vtip;

        optmu=.0107*(Vinf/1.69)-1.2;


        % if desired, iterate omega to obtain desired mu value
        if helochoice==2
           if rvrCP==1
              if mu < 1.0
                 clc
                 disp('The advance ratio, mu, is'),disp(mu)
                 disp('')
                 flag=1;
              while flag > 0
                 disp('This value is too low for RVR operation, do you want JANRAD to optimize mu so
the')
                 disp('rotor system will operate in RVR mode?')
                 disp('1. continue with current mu value or 2. change mu by automatically varying
rotational speed');
                 muchoice=input('Enter a 1 or 2...');
                    while isempty(muchoice),
                       disp(' ')
                       disp('You must enter a 1 or 2')
                       disp('Do you want JANRAD to optimize mu so the rotor system will operate in RVR
mode? ')
                       disp('1. continue with current mu value or 2. change mu by automatically varying
rotational speed');
                       muchoice=input('Enter a 1 or 2...');
                       disp('Optimum mu value currently set to'),disp(optmu)
                    end
                    if muchoice~=1 & muchoice~=2
                       disp(' ')
                       disp(' *** Enter a 1 or 2 ***')
                    elseif muchoice==2
                       muerror=mu-optmu;
                       if muerror < 0
                          while abs(muerror) > .01
                             omega=omega - .001;
                             Vtip=omega*R;
                             mu=Vinf*cos(alphaT)/Vtip;
                             muerror=mu-optmu;
                          end
                          muerror=0;
                          disp('New rotational velocity is '),disp(omega)
                          disp('')
                          disp('New mu value is '),disp(mu)
                          disp('')
                          disp('Press any key to continue ...')
                          pause
                       else
                       end
                       flag=0;
                    elseif muchoice==1
                       disp('Rotational velocity and mu value unchanged.')
```

```
                        disp('')
                        disp('Press any key to continue ...')
                        pause
                        flag=0;
                    end
                end
            else
            end
        else
        end
    else
    end
    if mu <= 1.0 & rvrCP==1
        clc
        disp('The advance ratio is less than 1.0 and you have chosen RVR based')
        disp('cyclic pitch.  JANRAD cannot complete computations using this input')
        disp('The program will revert to conventional Higher Harmonic Feathering.')
        disp('Press any key to continue ...')
        pause
        rvrCP=2;
    else
    end

    if rvrCP==1 & twoPinput==0
        clc
        disp('You have selected RVR cyclic pitch, 2/rev cyclic input cannot be zero.')
        twoPinput=input('Please enter a non-zero 2/rev cyclic input in degrees or press <CONTROL +
C> to quit JANRAD');

        while isempty(twoPinput),
            disp(' ')
            disp('You must enter a value')
            twoPinput=input('2/rev cyclic input (degs): ');
            while twoPinput==0
                disp('The value cannot be zero either!')
                twoPinput=input('Please enter a non-zero 2/rev cyclic input or press <CONTROL + C> to
quit JANRAD');
            end
        end

    else
    end

    clc
    disp('          *** ANALYSIS IN PROGRESS ***')
    %   *** setup blade radius elements, azimuth elements,
    %       induced velocity distributions, and determination
    %       of coning angle and tip loss parameters ***

    B=1-(sqrt(2*CT)/b);
    Reff=B*R;
    Rbar=Reff-e;
    dr=(Reff-grip)/nbe;
    r=grip:dr:Reff-dr;,r=r+dr/2;
    rT1=.7;,%   *** first guess at rT ***
    RbarT=rT1*Rbar;
```

```
            mblade=wblade/32.17;
            betao=asin((T/b*RbarT-(.5*(R-e)+e)*wblade)/((.5*(R-e)+e)^2*omega^2*mblade));
            betat=twist*(0.7-(r/R));
            psi=0:360/naz:360-360/naz;,psi=psi'/57.3;

            %   *** induced velocity distribution ***

            if Vinf < 16.9,
               A=4*pi;
               Bv=(b/2)*omega*a*cblade;
               Tv=0;
               delT=T-Tv;
               while abs(delT) > .01*T
                  thetav=twist*(0.7-(r/R))+thetao;
                  C=(-b/2)*cblade*omega^2*r*a.*thetav;
                  vi=(-Bv+sqrt(Bv^2-(4*A*C)))/(2*A);
                  dTv=(b/2)*rho*((omega*r).^2)*a.*(thetav-(vi./(omega*r)))*cblade*dr;
                  Tv=sum(dTv);
                  delT=T-Tv;
                  if delT < 0,
                     thetao=thetao-0.5*thetao*abs(delT/T);
                  else
                     thetao=thetao+0.5*thetao*abs(delT/T);
                  end
               end
            else
               lamdaT=[1-2*mu*sin(alphaT)                              mu^2*(sin(alphaT)^2+1)-2*mu^3*sin(alphaT)
mu^4*sin(alphaT)^2-CT^2/4];
               lamdaT=max(real(roots(lamdaT)));
               vi=lamdaT*Vtip-Vinf*sin(alphaT);
               vi=vi*ones(size(r));
            end

            %  *** first guess at theta ***

            theta1c=0.035*((0.0006e-3*Vinf^2+0.244e-3*Vinf)/0.105);
            theta1s=-0.087*((0.0006e-3*Vinf^2+0.244e-3*Vinf)/0.105);
            theta2c=twoPinput/57.3;
            theta3s=threePinput/57.3;
            theta3Poffset=threePoffset/57.3;

            theta=thetao+theta1c.*cos(psi)+theta1s.*sin(psi);
            theta2Pr=theta2c.*cos(2*psi);
            theta3Pr=theta3s.*sin(3*psi+theta3Poffset);
            thetaXP=thetao+theta1c.*cos(psi)+theta1s.*sin(psi)+theta2c.*cos(2*psi)+theta3s.*sin(3*psi+theta
3Poffset);
            if rvrCP==1
               thetaXP=thetao+thetao*sin(psi)+theta1c.*cos(psi)+theta1s.*sin(psi)+theta2c.*cos(2*psi);
               thetaXP=thetao+thetao*sin(psi)-theta1c.*cos(psi)-theta1s.*sin(psi)+theta2c.*cos(2*psi);
            else
            end

            %   *** rotor trimming routine ***

            disp(' ')
            disp(' ')
```

```
          disp('         *** TRIMMING COLLECTIVE ***')

          k=1;

          if rvrCP==1
             devTC=.20;
             devTC=.1;
          else
             devTC=.02;
          end

          error0=(T*devTC)+1;
          while abs(error0) > T*devTC;
             Tpsi=zeros(size(psi));
             thrcalc
             error0=T-(mean(Tpsi)*b);

             if error0 < -T*devTC
                thetao=thetao-0.35*thetao*abs(1.5*error0/T)*abs(1-mu);% added abs for RVR operation
             elseif error0 > T*devTC
                thetao=thetao+0.35*thetao*abs(1.5*error0/T)*abs(1-mu);% added abs for RVR operation
             end

             theta=thetao+theta1c.*cos(psi)+theta1s.*sin(psi);
             theta2Pr=theta2c.*cos(2*psi);
             theta3Pr=theta3s.*sin(3*psi+theta3Poffset);

thetaXP=thetao+theta1c.*cos(psi)+theta1s.*sin(psi)+theta2c.*cos(2*psi)+theta3s.*sin(3*psi+theta3Poffset)
;
             if rvrCP==1
                thetaXP=thetao+thetao*sin(psi)+theta1c.*cos(psi)+theta1s.*sin(psi)+theta2c.*cos(2*psi);
                thetaXP=thetao+thetao*sin(psi)-theta1c.*cos(psi)-theta1s.*sin(psi)+theta2c.*cos(2*psi);
             else
             end
             if k > 1,
                if abs(error0) >= abs(error1),
                   clc
                   disp(' ')
                   disp(' ')
                   disp('This configuration will not trim')
                   disp('Try a lower airspeed or a new design')
                   disp(' ')
                   disp('Program execution terminated')
                   disp(' ')
                   error('*** END OF PROGRAM ***')
                end
             end
             error1=error0;
             if rvrCP==1
                error0=(T*devTC)-1;
             end
          end

          disp(' ')
          disp('         *** TRIMMING CYCLIC ***')
          t0=clock;
```

```
k=1;

if rvrCP==2
   error0=(((T/b)*rT1*(R-grip))*.04)+1;
   while error0 > ((T/b)*rT1*(R-grip))*.04
      time=etime(clock,t0);

      if time > 15,
         disp(' ')
         disp('still trimming')
         t0=clock;
      end

      Mpsi(:,k)=zeros(size(psi));
      tmcalc
      theta=[theta theta(:,k)];
      theta2Pr=[theta2Pr theta2Pr(:,k)];
      theta3Pr=[theta3Pr theta3Pr(:,k)];
      thetaXP=[thetaXP thetaXP(:,k)];
      Mpsi=[Mpsi Mpsi(:,k)];
%   *** calculation of initial dthetadM ***

      if k < 2,
         theta(:,k+1)=theta(:,k)+0.25/57.3;
         theta2Pr(:,k+1)=theta2Pr(:,k)+0.25/57.3;
         thetaXP(:,k+1)=thetaXP(:,k)+0.25/57.3;
         Mpsi(:,k+1)=zeros(size(psi));
         k=k+1;
         tmcalc
         k=k-1;
         dthetadM=(thetaXP(:,k+1)-thetaXP(:,k))./(Mpsi(:,k+1)-Mpsi(:,k));
      end

%   *** calculation of M first harmonic parameters ***

      M1c=2*sum(Mpsi(:,k).*cos(psi))/naz;
      M1s=2*sum(Mpsi(:,k).*sin(psi))/naz;

%   *** removal of first harmonic terms from Mpsi ***

      Mpsi(:,k+1)=Mpsi(:,k)-M1c.*cos(psi)-M1s.*sin(psi);
      delM=Mpsi(:,k+1)-Mpsi(:,k);
      error0=max(delM)-min(delM);
      if k > 1,
         if error0 > error1,
            clc
            disp(' ')
            disp(' ')
            disp('This configuration will not trim')
            disp('Try a lower airspeed or a new design')
            disp(' ')
            disp('Program execution terminated')
            disp(' ')
            error('*** END OF PROGRAM ***')
         end
      end
```

```
            error1=error0;

        %   *** calculation of new theta ***

            delM=0.5*(1-mu)*delM;
            theta(:,k+1)=theta(:,k)+(dthetadM.*delM);
            theta2Pr(:,k+1)=theta2Pr(:,k)+(dthetadM.*delM);
            theta3Pr(:,k+1)=theta3Pr(:,k)+(dthetadM.*delM);
            thetaXP(:,k+1)=thetaXP(:,k)+(dthetadM.*delM);

            if error0 <= ((T/b)*rT1*(R-grip))*.04,
               theta1c=2*sum(theta(:,k).*cos(psi))/naz;
               theta1s=2*sum(theta(:,k).*sin(psi))/naz;
        %   *** 2P coefficient ***
               theta2c=sum(ones(size(theta(:,k))).*twoPinput/57.3)/naz;
        %   *** 3P coeeficient ***
               theta3s=sum(ones(size(theta(:,k))).*threePinput/57.3)/naz;
            else
               theta1c=2*sum(theta(:,k+1).*cos(psi))/naz;
               theta1s=2*sum(theta(:,k+1).*sin(psi))/naz;
        %   *** 2P coefficient ***
               theta2c=sum(ones(size(theta(:,k+1))).*twoPinput/57.3)/naz;
        %   *** 3P coefficient ***
               theta3s=sum(ones(size(theta(:,k+1))).*threePinput/57.3)/naz;
            end

        %   *** calculate theta with 1P terms only ***
            theta(:,k+1)=thetao+theta1c.*cos(psi)+theta1s.*sin(psi);

        %   *** add 2P and 3P terms, result thetaXP ***
            theta2Pr(:,k+1)=theta2c.*cos(2*psi);
            theta3Pr(:,k+1)=theta3s.*sin(3*psi+theta3Poffset);

thetaXP(:,k+1)=thetao+theta1c.*cos(psi)+theta1s.*sin(psi)+theta2c.*cos(2*psi)+theta3s.*sin(3*psi+theta3
Poffset);
            if rvrCP==1

thetaXP(:,k+1)=thetao+thetao*sin(psi)+theta1c.*cos(psi)+theta1s.*sin(psi)+theta2c.*cos(2*psi);
               thetaXP=thetao+thetao*sin(psi)-theta1c.*cos(psi)-theta1s.*sin(psi)+theta2c.*cos(2*psi);
            else
            end
        %   *** calculation of new dthetadM ***
            theta=[theta theta(:,k+1)];
            theta2Pr=[theta2Pr theta2Pr(:,k+1)];
            theta3Pr=[theta3Pr theta3Pr(:,k+1)];
            thetaXP=[thetaXP thetaXP(:,k+1)];

            Mpsi=[Mpsi Mpsi(:,k+1)];

            theta(:,k+2)=theta(:,k)+0.25/57.3;
            theta2Pr(:,k+2)=theta2Pr(:,k)+0.25/57.3;
            theta3Pr(:,k+2)=theta3Pr(:,k)+0.25/57.3;
            thetaXP(:,k+2)=thetaXP(:,k)+0.25/57.3;

            Mpsi(:,k+2)=zeros(size(Mpsi(:,k+1)));
            k=k+2;
        tmcalc
        k=k-2;
        dthetadM=(thetaXP(:,k+2)-thetaXP(:,k))./(Mpsi(:,k+2)-Mpsi(:,k));
        k=k+1;
    end
else
    Mpsi(:,k)=zeros(size(psi));

    tmcalc

    theta=[theta theta(:,k)];
    theta2Pr=[theta2Pr theta2Pr(:,k)];
    theta3Pr=[theta3Pr theta3Pr(:,k)];
    thetaXP=[thetaXP thetaXP(:,k)];

    Mpsi=[Mpsi Mpsi(:,k)];

%    *** calculation of initial dthetadM ***

    if k < 2,
          theta(:,k+1)=theta(:,k)+0.25/57.3;
          theta2Pr(:,k+1)=theta2Pr(:,k)+0.25/57.3;
          thetaXP(:,k+1)=thetaXP(:,k)+0.25/57.3;
          Mpsi(:,k+1)=zeros(size(psi));
          k=k+1;
          tmcalc
          k=k-1;
          dthetadM=(thetaXP(:,k+1)-thetaXP(:,k))./(Mpsi(:,k+1)-Mpsi(:,k));
    end
end

disp(' ')
disp('          *** ADJUSTING COLLECTIVE ***')
disp(' ')

theta=theta(:,k);
theta2Pr=theta2Pr(:,k);
theta3Pr=theta3Pr(:,k);
thetaXP=thetaXP(:,k);

k=1;

if rvrCP==1
    devAC=.1;
    devAC=.03;
else
    devAC=.01;
end

error0=(T*devAC)+1;
while abs(error0) > T*devAC
    Tpsi=zeros(size(psi));
    thrcalc
    error0=T-(mean(Tpsi)*b);
    if error0 < -T*devAC,
        thetao=thetao-0.25*thetao*abs(1.25*error0/T)*abs(1-mu);
```

```
                elseif error0 > T*devAC,
                    thetao=thetao+.025*thetao*abs(1.25*error0/T)*abs(1-mu);
                end
                    theta=thetao+theta1c.*cos(psi)+theta1s.*sin(psi);
                    theta2Pr=theta2c.*cos(2*psi);
                    theta3Pr=theta3s.*sin(3*psi+threePoffset);

thetaXP=thetao+theta1c.*cos(psi)+theta1s.*sin(psi)+theta2c.*cos(2*psi)+theta3s.*sin(3*psi+theta3Poffset)
;
                if rvrCP==1
                    thetaXP=thetao+thetao*sin(psi)+theta1c.*cos(psi)+theta1s.*sin(psi)+theta2c.*cos(2*psi);
                    thetaXP=thetao+thetao*sin(psi)-theta1c.*cos(psi)-theta1s.*sin(psi)+theta2c.*cos(2*psi);
                else
                end
                if k > 1,
                    if abs(error0) > abs(error1),
                        clc
                        disp(' ')
                        disp(' ')
                        disp('This configuration will not trim')
                        disp('Try a lower airspeed or a new design')
                        disp(' ')
                        disp('Program execution terminated')
                        disp(' ')
                        error('END OF PROGRAM')
                    end
                end
                error1=error0;
                k=k+1;
                if rvrCP==1
                    error0=(T*devAC)-1;
                end
            end

            %   *** calculating drag moments ***

            DMpsi=zeros(size(psi));
            dmcalc

            %   *** calculating rotor H force ***

            if Vinf < 16.9,
                Hrotor=0;
                dT=[dT ddT];
                dN=[dN ddN];
                dD=[dD ddD];
            else
                dT=[dT ddT];
                dN=[dN ddN];
                dD=[dD ddD];
                for i=1:length(r)+1,
                    H1c(i)=2*sum(dT(:,i).*cos(psi))/naz;
                    H1s(i)=2*sum(dT(:,i).*sin(psi))/naz;
                end
                if rvrCP==2
                    Hrotor=(((b*cos(alphaT)/2)*(sum(H1s)-sin(betao)*sum(H1c)))+Drotor)/2;
```

```
    else
        Hrotor=0;
    end
end

%    *** calculating new rT ***

rT2=(((mean(Mpsi(:,length(Mpsi(1,:))-1))/mean(Tpsi))/R)+rT1)/2;

%    *** check rotor drag and rT, retrim rotor if needed ***


while abs(Drotor-Hrotor) > 0.2*Hrotor | abs(rT1-rT2) > 0.5*rT1 % increase tolerance to 50%
    if abs(Drotor-Hrotor) > 0.2*Hrotor,
        disp(' ')
        disp('         *** ADJUSTING ROTOR DRAG ***')
    end

    if rvrCP==1
        Drotor=0;
    else
        Drotor=Hrotor;
    end

    if rvrCP==1
        devMT=.25;
        devMT=.045;
    else
        devMT=.015;
    end

    if abs(rT1-rT2) > devMT*rT1,
        disp(' ')
        disp('         *** ADJUSTING MEAN THRUST LOCATION ***')
    end

    disp(' ')
    disp('         *** RETRIMMING ROTOR ***')
    disp(' ')
    dT=dT(:,1:nbe);
    dN=dN(:,1:nbe);
    dD=dD(:,1:nbe);

%   *** recealculating parameters ***

%   if required, define alphaT fixed quantity
    if alphaTfix==1
        alphaT=alphaTinput;
    else
        alphaT=atan((Dftotal+Drotor)/(GW-Lftotal));
    end

    mu=Vinf*cos(alphaT)/Vtip;

    if Vinf >= 16.9,
        T=(GW-Lftotal)/cos(alphaT);
```

```
            CT=T/(Adisk*rho*Vtip^2);
            lamdaT=[1          -2*mu*sin(alphaT)          mu^2*(sin(alphaT)^2+1)-2*mu^3*sin(alphaT)
mu^4*sin(alphaT)^2-CT^2/4];
            lamdaT=max(real(roots(lamdaT)));
            vi=lamdaT*Vtip-Vinf*sin(alphaT);
            vi=vi*ones(size(r));
          end

          B=1-(sqrt(2*CT)/b);
          Reff=B*R;
          Rbar=Reff-e;
          dr=(Reff-grip)/nbe;
          r=grip:dr:Reff-dr;,r=r+dr/2;
          RbarT=rT2*Rbar;
          betao=asin((T/b*RbarT-(.5*(R-e)+e)*wblade)/((.5*(R-e)+e)^2*omega^2*mblade));

        %  *** trimming collective ***

          t0=clock;
          k=1;
          error0=(T*devTC)+1;

          while abs(error0) > T*devTC
            Tpsi=zeros(size(psi));
            thrcalc
            error0=T-(mean(Tpsi)*b);
            if error0 < -T*devTC,
              thetao=thetao-0.35*thetao*abs(1.5*error0/T)*abs(1-mu);
            elseif error0 > T*devTC,
              thetao=thetao+0.35*thetao*abs(1.5*error0/T)*abs(1-mu);
            end
            theta=thetao+theta1c.*cos(psi)+theta1s.*sin(psi);
            theta2Pr=theta2c.*cos(2*psi);
            theta3Pr=theta3s.*sin(3*psi+threePoffset);

thetaXP=thetao+theta1c.*cos(psi)+theta1s.*sin(psi)+theta2c.*cos(2*psi)+theta3s.*sin(3*psi+theta3Poffset)
;
            if rvrCP==1
              thetaXP=thetao+thetao*sin(psi)+theta1c.*cos(psi)+theta1s.*sin(psi)+theta2c.*cos(2*psi);
              thetaXP=thetao+thetao*sin(psi)-theta1c.*cos(psi)-theta1s.*sin(psi)+theta2c.*cos(2*psi);
            else
            end
            if k > 1,
              if abs(error0) > abs(error1),
                clc
                disp(' ')
                disp(' ')
                disp('This configuration will not trim')
                disp('Try a lower airspeed or a new design')
                disp(' ')
                disp('Program execution terminated at line 746')
                disp(' ')
                error('*** END OF PROGRAM ***')
              end
            end
            error1=error0;
        k=k+1;
        if rvrCP==1
            error0=(T*devTC)-1;
        end
    end

%   *** trimming cyclic ***

    if rvrCP==1
        devCY=.2;
        devCY=.12;
    else
        devCY=.04;
    end

    k=1;
    error0=(((T/b)*rT2*(R-grip))*devCY)+1;

    if rvrCP==2
        while error0 > ((T/b)*rT2*(R-grip))*devCY
            time=etime(clock,t0);
            if time > 15,
                disp('still trimming...')
                disp(' ')
                t0=clock;
            end
            Mpsi(:,k)=zeros(size(psi));
            tmcalc
            theta=[theta theta(:,k)];
            theta2Pr=[theta2Pr theta2Pr(:,k)];
            theta3Pr=[theta3Pr theta3Pr(:,k)];
            thetaXP=[thetaXP thetaXP(:,k)];
            Mpsi=[Mpsi Mpsi(:,k)];

%   *** calculation of initial dthetadM ***

            if k < 2,
                theta(:,k+1)=theta(:,k)+0.25/57.3;
                theta2Pr(:,k+1)=theta2Pr(:,k)+0.25/57.3;
                theta3Pr(:,k+1)=theta3Pr(:,k)+0.25/57.3;
                thetaXP(:,k+1)=thetaXP(:,k)+0.25/57.3;
                Mpsi(:,k+1)=zeros(size(psi));
                k=k+1;
                tmcalc
                k=k-1;
                dthetadM=(thetaXP(:,k+1)-thetaXP(:,k))./(Mpsi(:,k+1)-Mpsi(:,k));
            end

%   *** calculation of M first harmonic parameters ***

            M1c=2*sum(Mpsi(:,k).*cos(psi))/naz;
            M1s=2*sum(Mpsi(:,k).*sin(psi))/naz;

%   *** removal of first harmonic terms from Mpsi ***

            Mpsi(:,k+1)=Mpsi(:,k)-M1c.*cos(psi)-M1s.*sin(psi);
            delM=Mpsi(:,k+1)-Mpsi(:,k);
            error0=max(delM)-min(delM);
            if k > 1,
               if error0 > error1,
                  clc
                  disp(' ')
                  disp('This configuration will not trim')
                  disp('Try a lower airspeed or a new design')
                  disp(' ')
                  disp('Program Execution Terminated at line 814')
                  disp(' ')
                  error('*** END OF PROGRAM ***')
               end
            end
            error1=error0;

      %   *** calculation of new theta ***

            delM=0.5*(1-mu)*delM;
            theta(:,k+1)=theta(:,k)+(dthetadM.*delM);
            theta2Pr(:,k+1)=theta2Pr(:,k)+(dthetadM.*delM);
            theta3Pr(:,k+1)=theta3Pr(:,k)+(dthetadM.*delM);
            thetaXP(:,k+1)=thetaXP(:,k)+(dthetadM.*delM);

            if error0 <= ((T/b)*rT2*(R-grip))*.04,
               theta1c=2*sum(theta(:,k).*cos(psi))/naz;
               theta1s=2*sum(theta(:,k).*sin(psi))/naz;
      %   2P/3P coefficients
               theta2c=sum(ones(size(theta(:,k))).*twoPinput/57.3)/naz;
               theta3s=sum(ones(size(theta(:,k))).*threePinput/57.3)/naz;
            else
               theta1c=2*sum(theta(:,k+1).*cos(psi))/naz;
               theta1s=2*sum(theta(:,k+1).*sin(psi))/naz;
      %   2P/3P coefficients
               theta2c=sum(ones(size(theta(:,k+1))).*twoPinput/57.3)/naz;
               theta3s=sum(ones(size(theta(:,k+1))).*threePinput/57.3)/naz;
            end

      %   calculate theta with 1P terms only
            theta(:,k+1)=thetao+theta1c.*cos(psi)+theta1s.*sin(psi);

      %   add 2P and 3P terms, unit value only for theta2c, result thetaXP
            theta2Pr(:,k+1)=theta2c.*cos(2*psi);
            theta3Pr(:,k+1)=theta3s.*sin(3*psi+threePoffset);

thetaXP(:,k+1)=thetao+theta1c.*cos(psi)+theta1s.*sin(psi)+theta2c.*cos(2*psi)+theta3s.*sin(3*psi+theta3
Poffset);

            if rvrCP==1

thetaXP(:,k+1)=thetao+thetao*sin(psi)+theta1c.*cos(psi)+theta1s.*sin(psi)+theta2c.*cos(2*psi);
               thetaXP(:,k+1)=thetao+thetao*sin(psi)-theta1c.*cos(psi)-
theta1s.*sin(psi)+theta2c.*cos(2*psi);
            else
            end
       %   *** calculation of new dthetadM ***
```

```
            theta=[theta theta(:,k+1)];
            theta2Pr=[theta2Pr theta2Pr(:,k+1)];
            theta3Pr=[theta3Pr theta3Pr(:,k+1)];
            thetaXP=[thetaXP thetaXP(:,k+1)];
            Mpsi=[Mpsi Mpsi(:,k+1)];
            theta(:,k+2)=zeros(size(Mpsi(:,k+1)));
            theta2Pr(:,k+2)=zeros(size(Mpsi(:,k+1)));
            theta3Pr(:,k+2)=zeros(size(Mpsi(:,k+1)));
            thetaXP(:,k+2)=zeros(size(Mpsi(:,k+1)));

            k=k+2;
            tmcalc
            k=k-2;

%   replace 1P dthetadM with 2P dthetadM
%   dthetadM=(theta(:,k+2)-theta(:,k))./(Mpsi(:,k+2)-Mpsi(:,k));
            dthetadM=(thetaXP(:,k+2)-thetaXP(:,k))./(Mpsi(:,k+2)-Mpsi(:,k));
            k=k+1;
        end

    else
        Mpsi(:,k)=zeros(size(psi));
        tmcalc
        theta=[theta theta(:,k)];
        theta2Pr=[theta2Pr theta2Pr(:,k)];
        theta3Pr=[theta3Pr theta3Pr(:,k)];
        thetaXP=[thetaXP thetaXP(:,k)];
        Mpsi=[Mpsi Mpsi(:,k)];

%   *** calculation of initial dthetadM ***

        if k < 2,
            theta(:,k+1)=theta(:,k)+0.25/57.3;
            theta2Pr(:,k+1)=theta2Pr(:,k)+0.25/57.3;
            theta3Pr(:,k+1)=theta3Pr(:,k)+0.25/57.3;
            thetaXP(:,k+1)=thetaXP(:,k)+0.25/57.3;
            Mpsi(:,k+1)=zeros(size(psi));
            k=k+1;
            tmcalc
            k=k-1;
            dthetadM=(thetaXP(:,k+1)-thetaXP(:,k))./(Mpsi(:,k+1)-Mpsi(:,k));
        end
    end

%   *** retrimming collective ***

    theta=theta(:,k);
    theta2Pr=theta2Pr(:,k);
    theta3Pr=theta3Pr(:,k);
    thetaXP=thetaXP(:,k);
    k=1;
    error0=(T*devAC)+1;
    while abs(error0) > T*devAC
        Tpsi=zeros(size(psi));
        thrcalc
```

```
                error0=T-(mean(Tpsi)*b);
                if error0 < -T*devAC,
                   thetao=thetao-0.25*thetao*abs(1.25*error0/T)*abs(1-mu);
                elseif error0 > T*devAC,
                   thetao=thetao+0.25*thetao*abs(1.25*error0/T)*abs(1-mu);
                end
                theta=thetao+theta1c.*cos(psi)+theta1s.*sin(psi);
                theta2Pr=theta2c.*cos(2*psi);
                theta3Pr=theta3s.*sin(3*psi+threePoffset);

thetaXP=thetao+theta1c.*cos(psi)+theta1s.*sin(psi)+theta2c.*cos(2*psi)+theta3s.*sin(3*psi+theta3Poffset)
;
                if rvrCP==1
                   thetaXP=thetao+thetao*sin(psi)+theta1c.*cos(psi)+theta1s.*sin(psi)+theta2c.*cos(2*psi);
                   thetaXP=thetao+thetao*sin(psi)-theta1c.*cos(psi)-theta1s.*sin(psi)+theta2c.*cos(2*psi);
                else
                end
                if k > 1,
                   if abs(error0) > abs(error1),
                      clc
                      disp(' ')
                      disp('This configuration will not trim')
                      disp('Try a lower airspeed or a nex design')
                      disp(' ')
                      error('END OF PROGRAM at line 933')
                   end
                end
                error1=error0;
                k=k+1;
                if rvrCP==1
                   error0=(T*devAC)-1;
                end
             end

          %   *** recalculating rotor H force ***

             if Vinf < 16.9,
                Hrotor=0;
                dT=[dT ddT];
                dN=[dN ddN];
                dD=[dD ddD];
             else
                dT=[dT ddT];
                dN=[dN ddN];
                dD=[dD ddD];
                for i=1:length(r)+1,
                   H1c(i)=2*sum(dT(:,i).*cos(psi))/naz;
                   H1s(i)=2*sum(dD(:,i).*sin(psi))/naz;
                end

                if rvrCP==1
                   Hrotor=0;
                else
                   Hrotor=(((b*cos(alphaT)/2)*(sum(H1s)-sin(betao)*sum(H1c)))+Drotor)/2;
                end
```

```
    end

%   *** recalculating rT ***

    rT1=rT2;
    rT2=(((mean(Mpsi(:,length(Mpsi(1,:))-1))/mean(Tpsi))/R)+rT1)/2;

end

%   *** recalculating drag moments ***

dT=dT(:,1:nbe);
dN=dN(:,1:nbe);
dD=dD(:,1:nbe);
DMpsi=zeros(size(psi));
dmcalc
dT=[dT ddT];
dN=[dN ddN];
dD=[dD ddD];
clc
disp(' ')
disp('          *** ROTOR TRIMMED ***')
disp('       press any key to continue...')
pause
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX I. OO12CLCD.M

```
function [CL,CD]=oo12clcd(alpha, mach)
%
% oo12clcd calculates CL and CD for the NACA 0012
% airfoil given angle of attack in radians and the
% local Mach number:
%
% [CL,CD]=oo12clcd(alpha, mach)
%
% Both 'alpha' and 'mach' are intended to be vectors
% the elements of which correspond to the rotor blade
% radial stations of interest in a blade element analysis.
% All equations are based on Ray Prouty's treatment of
% the 0012 in his text.


CL=zeros(size(alpha));
CD=zeros(size(alpha));
a=alpha*180/pi;
mach=abs(mach);
aL = 15 - 16.*mach;
aD = 17 - 23.4.*mach;
K1 = 0.0233 + 0.342.*(mach.^7.15);
K2 = 2.05 - 0.95.*mach;


% CL for Mach numbers < 0.725 and AOA inside +/- 20 deg:

chk=(mach<0.725 & a>=0 & a<=aL);
 CL=CL+chk.*((0.1./sqrt(1-mach.^2) - 0.01.*mach).*a);

chk=(mach<0.725 & a>aL & a<=20);
 CL=CL+chk.*((0.1./sqrt(1-mach.^2) - 0.01.*mach).*a - K1.*(a-aL).^K2);

chk=(mach<0.725 & a>=-20 & a<-aL);
 CL=CL-chk.*((0.1./sqrt(1-mach.^2) - 0.01.*mach).*abs(a) - K1.*(abs(a)-aL).^K2);

chk=(mach<0.725 & a>=-aL & a<0);
 CL=CL-chk.*((0.1./sqrt(1-mach.^2) - 0.01.*mach).*abs(a));



% CL for Mach numbers > 0.725 and AOA inside +/- 20 deg:

chk=(mach>=0.725 & a>=0 & a<=aL);
 CL=CL+chk.*((0.677 - 0.744.*mach).*a);

chk=(mach>=0.725 & a>aL & a<=20);
 CL=CL+chk.*((0.677 - 0.744.*mach).*a - (0.0575-0.144.*(mach-0.725).^0.44).*(a-aL).^(K2));

chk=(mach>=0.725 & a<0 & a>=-aL);
 CL=CL-chk.*((0.677 - 0.744.*mach).*abs(a));
```

```
chk=(mach>=0.725 & a<-aL & a>=-20);
 CL=CL-chk.*((0.677 - 0.744.*mach).*abs(a) - (0.0575-0.144.*(mach-0.725).^0.44).*(abs(a)-aL).^(K2));


% CL for all Mach numbers and AOA outside +/- 20deg:

chk=(a>20 & a<=161);
 CL=CL+chk.*(1.15.*sin(2.*alpha));

chk=(a>161 & a<=173);
 CL=CL+chk.*(-0.7);

chk=(a>173 & a<=180);
 CL=CL+chk.*(0.1.*(a-180));

chk=(a>=-180 & a<=-173);
 CL=CL+chk.*(0.1.*(a+180));

chk=(a>-173 & a<=-161);
 CL=CL+chk.*(0.7);

chk=(a>-161 & a<-20);
 CL=CL+chk.*(1.15.*sin(2.*alpha));



% CD for Mach numbers < 0.725 and AOA inside +/- 20 deg:

chk=(mach<0.725 & a>=0 & a<=aD);
 CD=CD+chk.*(0.0081 + (-350.*a + 396.*a.^2 - 63.3.*a.^3 + 3.66.*a.^4).*10.^(-6));

chk=(mach<0.725 & a>aD & a<=20);
 CD=CD+chk.*((0.0081 + (-350.*a + 396.*a.^2 - 63.3.*a.^3 + 3.66.*a.^4).*10.^(-6)) + 0.00066.*(a-aD).^2.54);

chk=(mach<0.725 & a<0 & a>=-aD);
 CD=CD+chk.*(0.0081 + (-350.*abs(a) + 396.*a.^2 - 63.3.*abs(a).^3 + 3.66.*a.^4).*10.^(-6));

chk=(mach<0.725 & a<-aD & a>=-20);
 CD=CD+chk.*((0.0081 + (-350.*abs(a) + 396.*a.^2 - 63.3.*abs(a).^3 + 3.66.*a.^4).*10.^(-6)) + 0.00066.*(abs(a)-aD).^2.54);



% CD for Mach numbers > 0.725 and AOA inside +/- 20 deg:

chk=(mach>=0.725 & a>=0 & a<=20);
 CD=CD+chk.*((0.0081 + (-350.*a + 396.*a.^2 - 63.3.*a.^3 + 3.66.*a.^4).*10.^(-6)) + 0.00035.*a.^2.54 + 21.*(mach-0.725).^3.2);

chk=(mach>=0.725 & a<0 & a>=-20);
 CD=CD+chk.*((0.0081 + (-350.*abs(a) + 396.*a.^2 - 63.3.*abs(a).^3 + 3.66.*a.^4).*10.^(-6)) + 0.00035.*abs(a).^2.54 + 21.*(mach-0.725).^3.2);
```

```
% CD for all Mach numbers and AOA outside +/- 20deg:

chk=(a>20 & a<=180);
 CD=CD+chk.*(1.03 - 1.02.*cos(2.*alpha));

chk=(a>=-180 & a<-20);
 CD=CD+chk.*(1.03 - 1.02.*cos(2.*alpha));
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX J. HH02CLCD.M

```
function [CL,CD]=hh02clcd(alpha)
% hh02clcd calculates CL and CD for an HH-02 airfoil
% given angle of attack (alpha) in radians and Mach number (Mach)
% [CL,CD]=hh02clcd(alpha,mach)
CL=zeros(size(alpha));
CD=zeros(size(alpha));
a=alpha*180/pi;

chk1=(a>=20 & a<=180);
CL=CL+chk1.*(0.42541+0.026863*a+5.5988e-4*a.^2-2.1493e-5*a.^3+1.5932e-7*a.^4-3.4659e-
10*a.^5);
CD=CD+chk1.*(-0.7179+0.061213*a-5.9861e-4*a.^2+7.3708e-6*a.^3-6.6605e-8*a.^4+1.913e-
10*a.^5);

chk1=(a>=-180 & a<=-50);
CL=CL+chk1.*(-4.6183-0.1923*a-3.5554e-3*a.^2-3.3273e-5*a.^3-1.4528e-7*a.^4-2.3003e-
10*a.^5);
CD=CD+chk1.*(2.7093e-2-2.1309e-2*a+2.0335e-4*a.^2+3.47e-7*a.^3-3.0586e-8*a.^4-1.2584e-
10*a.^5);

chk1=(a>-50 & a<-20);
CL=CL+chk1.*(-2.5519-0.22847*a-9.5667e-3*a.^2-1.7051e-4*a.^3-1.0909e-6*a.^4);
CD=CD+chk1.*(2.7093e-2-2.1309e-2*a+2.0335e-4*a.^2+3.47e-7*a.^3-3.0586e-8*a.^4-1.2584e-
10*a.^5);

chk1=(a>=-20 & a<=-10);
CL=CL+chk1.*(-0.2+0.089*a+0.0034*a.^2);
CD=CD+chk1.*(2.7093e-2-2.1309e-2*a+2.0335e-4*a.^2+3.47e-7*a.^3-3.0586e-8*a.^4-1.2584e-
10*a.^5);

chk1=(a<20 & a>-10);
CL=CL+chk1.*(5.8766e-2+1.3131e-1*a+2.4742e-3*a.^2-5.303e-4*a.^3-1.5818e-5*a.^4+1.28e-
6*a.^5);
   chk2=a<-4;
   chk2=chk2.*chk1;
CD=CD+chk2.*(1.3786+0.916*a+0.21396*a.^2+2.0371e-2*a.^3+7.0076e-4*a.^4);
   chk2=(a>=-4 & a<=7);
   chk2=chk2.*chk1;

CD=CD+chk2.*(9.732e-3+3.2326e-4*a+1.4392e-4*a.^2-8.5073e-5*a.^3+1.1826e-
6*a.^4+1.5271e-6*a.^5);
   chk2=a>7;
   chk2=chk2.*chk1;
   CD=CD+chk2.*(1.842e-1-5.7532e-2*a+5.8043e-3*a.^2-1.2803e-4*a.^3);
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX K. RVRCLCD.M

```
function [CL,CD]=rvrclcd(alpha,mach)
%RVRCLCD calculates CL and CD for a 12% thickness Fairchild
% RVR airfoil given angle of attack (alpha) in radians and
% mach number (mach).  Reference Fairchild Report HC144R1070
% figs D-28 and D-12

% JANRAD ver 1P2P3Pbeta

CL=zeros(size(alpha));
CD=zeros(size(alpha));

Mach_CL=[0 .26 .4 .6 .8 .9];
aoa_CL=[-180 -150 -120 -90 -30 -24 -20 -16 -12 -8 -4 0 4 8 12 16 20 24 30 90 120 150 160 170
180];
cl_tab=[    0       0       0       0       0       0;
            0       .7      .7      .7      .7      .7;
            0       .4      .4      .4      .4      .4;
            0       0       0       0       0       0;
            0       -.75    -.75    -.75    -.75    -.75;
            0       -.89    -.89    -.89    -.89    -.89;
            0       -.9     -.9     -.9     -.9     -.9;
            0       -1.0    -1.0    -1.0    -1.0    -1.0;
            0       -1.05   -1.05   -1.05   -1.05   -1.05;
            0       -.85    -.85    -.85    -.85    -.85;
            0       -.5     -.5     -.5     -.5     -.5;
            0       .25     .25     .25     .25     .25;
            0       .25     .25     .25     .25     .25;
            0       .7      .7      .7      .5      .55;
            0       .9      .9      .9      .7      .65;
            0       1.52    1.52    1.52    .78     1.0;
            0       1.65    1.65    1.65    .7      .95;
            0       1.5     1.5     1.5     1.2     1.2;
            0       1.7     1.7     1.7     1.7     1.7;
            0       0       0       0       0       0;
            0       -.55    -.55    -.55    -.55    -.55;
            0       -1.09   -1.09   -1.09   -1.09   -1.09;
            0       -.8     -.8     -.8     -.65    -.6;
            0       -1.1    -1.1    -1.1    -.7     -.5;
            0       0       0       0       0       0];

Mach_CD=[0 .26 .4 .6 .8 .9];
aoa_CD=[-180 -150 -120 -90 -30 -24 -20 -16 -12 -8 -4 0 4 8 12 16 20 24 30 90 120 150 160 170
180];
cd_tab=[    0       0       0       0       .02     .15;
            0       .65     .65     .65     .65     .65;
            0       1.2     1.2     1.2     1.2     1.2;
            0       1.65    1.65    1.65    1.65    1.65;
            0       .6      .6      .6      .6      .6;
            0       .5      .5      .5      .5      .5;
            0       .42     .42     .42     .42     .42;
            0       .4      .4      .4      .4      .4;
            0       .05     .05     .2      .2      .22;
```

```
          0     0     0     .15    .15    .2;
          0     0     0     0      .05    .12;
          0     0     0     0      .1     .1;
          0     0     0     0      .05    .12;
          0     0     0     .15    .15    .2;
          0     .05   .05   .2     .2     .22;
          0     .4    .4    .4     .4     .4;
          0     .42   .42   .42    .42    .42;
          0     .5    .5    .5     .5     .5;
          0     .6    .6    .6     .6     .6;
          0     1.65  1.65  1.65   1.65   1.65;
          0     1.2   1.2   1.2    1.2    1.2;
          0     .65   .65   .65    .65    .65;
          0     .45   .45   .45    .45    .45;
          0     .15   .15   .17    .2     .21;
          0     0     0     0      .05    .10];

rvra=alpha.*180/pi;
for j = 1:length(rvra)
   if rvra(j) < -180
      rvra(j) = rvra(j) + 360;
   elseif rvra(j) > 180
      rvra(j) = rvra(j) - 360;
   end
end
mach = abs(mach);
CL = interp2(Mach_CL,aoa_CL,cl_tab,mach,rvra);
CD = interp2(Mach_CD,aoa_CD,cd_tab,mach,rvra);
```

APPENDIX L. SC1095R8CLCD.M

```
function [CL,CD]=sc1095r8clcd(alpha,mach)
%sc1095r8 calculates CL and CD for a Sikorsky SC1095R8 airfoil
%given angle of attack (alpha) in radians and mach number (mach)
%[CL,CD]=sc1095r8clcd(alpha,mach)

CL=zeros(size(alpha));
CD=zeros(size(alpha));

Mach_CL=[0 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0];
aoa_CL=[-180 -178 -176 -174 -172 -170 -168 -166 -164 -162 -160 -158 -90 -22 -20 -18 -16 -14 -
12 -10 -8 ...
       -6 -4 -2 0 2 4 6 8 10 12 14 16 18 20 22 90 158 160 162 164 166 168 170 172 174 176 178
180];
cl_tab=[0.    0.    0.    0.    0.    0.    0.    0.    0.;
     .205  .205  .205  .205  .205  .205  .205  .205  .205;
     .41   .41   .41   .41   .41   .41   .41   .41   .41;
     .6    .6    .6    .6    .6    .6    .6    .6    .6;
     .77   .77   .77   .77   .77   .77   .77   .77   .77;
     .82   .82   .82   .82   .82   .82   .82   .82   .82;
     .82   .82   .82   .82   .82   .82   .82   .82   .82;
     .8    .8    .8    .8    .8    .8    .8    .8    .8;
     .76   .76   .76   .76   .76   .76   .76   .76   .76;
     .705  .705  .705  .705  .705  .705  .705  .705  .705;
     .65   .65   .65   .65   .65   .65   .65   .65   .65;
     .65   .65   .65   .65   .65   .65   .65   .65   .65;
    -.0627 -.0627 -.0627 -.0627 -.0627 -.0627 -.0627 -.0627 -.0627;
    -.98  -.98  -.98  -.914 -.934 -.926 -.875 -.838 -.822;
    -.975 -.975 -.96  -.910 -.93  -.920 -.856 -.81  -.79;
    -.969 -.969 -.962 -.906 -.926 -.914 -.838 -.782 -.758;
    -.963 -.963 -.966 -.902 -.922 -.908 -.819 -.754 -.726;
    -1.07 -1.07 -.824 -.803 -.805 -.88  -.8   -.726 -.694;
    -.718 -.718 -.532 -.528 -.66  -.83  -.79  -.698 -.662;
    -.366 -.366 -.24  -.4   -.61  -.78  -.81  -.67  -.630;
    -.245 -.245 -.3   -.33  -.55  -.74  -.75  -.666 -.622;
    -.39  -.39  -.44  -.32  -.52  -.68  -.69  -.663 -.615;
    -.4   -.4   -.42  -.44  -.47  -.58  -.47  -.486 -.428;
    -.185 -.185 -.185 -.196 -.199 -.255 -.25  -.31  -.24;
     .029  .029  .048  .048  .072  .07   .07   -.15  -.05;
     .244  .244  .282  .292  .343  .395  .35   .138  .2;
     .459  .459  .515  .536  .614  .72   .56   .39   .449;
     .673  .673  .749  .78   .84   .83   .705  .64   .7;
     .888  .888  .983  .96   .91   .882  .805  .765  .806;
    1.103 1.103 1.17  1.01  .946  .92   .842  .81   .85;
    1.25  1.25  1.13  .96   1.    .924  .845  .829  .865;
    1.1   1.1   1.03  1.07  1.053 .928  .848  .848  .88;
     .98   .98   .96   1.06  1.075 .92   .860  .867  .895;
     .982  .982  .966  1.07  1.064 .9    .880  .886  .91;
     .984  .984  .972  1.065 1.053 .9    .900  .905  .925;
     .987  .987  .979  1.06  1.042 .92   .920  .924  .94;
     .0627 .0627 .0627 .0627 .0627 .0627 .0627 .0627 .0627;
    -.66  -.66  -.66  -.66  -.66  -.66  -.66  -.66  -.66;
    -.655 -.655 -.655 -.655 -.655 -.655 -.655 -.655 -.655;
```

```
     -.685  -.685  -.685  -.685  -.685  -.685  -.685  -.685  -.685;
     -.73   -.73   -.73   -.73   -.73   -.73   -.73   -.73   -.73;
     -.77   -.77   -.77   -.77   -.77   -.77   -.77   -.77   -.77;
     -.8    -.8    -.8    -.8    -.8    -.8    -.8    -.8    -.8;
     -.805  -.805  -.805  -.805  -.805  -.805  -.805  -.805  -.805;
     -.79   -.79   -.79   -.79   -.79   -.79   -.79   -.79   -.79;
     -.61   -.61   -.61   -.61   -.61   -.61   -.61   -.61   -.61;
     -.42   -.42   -.42   -.42   -.42   -.42   -.42   -.42   -.42;
     -.21   -.21   -.21   -.21   -.21   -.21   -.21   -.21   -.21;
       0.     0.     0.     0.     0.     0.     0.     0.     0.];

Mach_CD=[0 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0];
aoa_CD=[-180 -178 -176 -174 -172 -170 -168 -166 -164 -162 -160 -158 -135 -90 -60 -45 -30 -22 -
20 -18 -16 -14 -12 -10 -8 ...
        -6 -4 -2 0 2 4 6 8 10 12 14 16 18 20 22 30 45 60 90 135 158 160 162 164 166 168 170 172
174 176 178 180];
cd_tab=[.02    .02    .02    .02    .02    .02    .02    .02    .02;
        .03    .03    .03    .03    .03    .03    .03    .03    .03;
        .05    .05    .05    .05    .05    .05    .05    .05    .05;
        .08    .08    .08    .08    .08    .08    .08    .08    .08;
        .11    .11    .11    .11    .11    .11    .11    .11    .11;
        .14    .14    .14    .14    .14    .14    .14    .14    .14;
       .185   .185   .185   .185   .185   .185   .185   .185   .185;
       .235   .235   .235   .235   .235   .235   .235   .235   .235;
       .25    .25    .25    .25    .25    .25    .25    .25    .25;
       .265   .265   .265   .265   .265   .265   .265   .265   .265;
       .295   .295   .295   .295   .295   .295   .295   .295   .295;
       .36    .36    .36    .36    .36    .36    .36    .36    .36;
      1.1945 1.1945 1.1945 1.1945 1.1945 1.1945 1.1945 1.1945 1.1945;
      2.022  2.022  2.022  2.022  2.022  2.022  2.022  2.022  2.022;
      1.662  1.662  1.662  1.662  1.662  1.662  1.662  1.662  1.662;
      1.194  1.194  1.194  1.194  1.194  1.194  1.194  1.194  1.194;
       .6     .6     .6     .6     .6     .6     .6     .6     .6;
       .3438  .3438  .3885  .4065  .414   .458   .479   .497   .514;
       .2723  .2723  .3281  .3506  .36    .415   .441   .463   .486;
       .2007  .2007  .2678  .2948  .3267  .372   .403   .43    .457;
       .1292  .1292  .2073  .2388  .2887  .329   .3655  .397   .428;
       .0576  .0576  .147   .183   .246   .286   .3278  .363   .399;
       .0174  .0174  .0225  .12    .191   .243   .29    .33    .37;
       .008   .008   .0132  .068   .127   .177   .225   .262   .297;
       .0082  .0082  .0095  .0206  .07    .113   .16    .203   .248;
       .0079  .0079  .0085  .0097  .026   .06    .1     .149   .202;
       .0075  .0075  .008   .008   .0125  .03    .065   .115   .152;
       .0075  .0075  .008   .0075  .0085  .012   .028   .066   .117;
       .0075  .0075  .008   .0075  .008   .008   .017   .05    .09;
       .008   .008   .0082  .0075  .0075  .0105  .04    .08    .1175;
       .0085  .0085  .0085  .008   .011   .036   .09    .12    .1525;
       .009   .009   .0105  .011   .029   .081   .128   .167   .203;
       .011   .011   .014   .026   .0743  .126   .17    .21    .249;
       .017   .017   .021   .08    .1247  .162   .225   .262   .298;
       .026   .026   .0935  .153   .18    .238   .285   .3225  .363;
       .145   .145   .1635  .2121  .246   .284   .326   .357   .393;
       .2147  .2147  .2259  .2643  .2887  .329   .3655  .391   .423;
       .274   .274   .2836  .3166  .3267  .327   .403   .43    .457;
       .3333  .3333  .3414  .3688  .36    .415   .441   .463   .486;
       .2927  .2927  .3991  .421   .414   .458   .479   .497   .514;
       .6    .6    .6    .6    .6    .6    .6    .6    .6;
      1.194  1.194  1.194  1.194  1.194  1.194  1.194  1.194  1.194;
      1.662  1.662  1.662  1.662  1.662  1.662  1.662  1.662  1.662;
      2.022  2.022  2.022  2.022  2.022  2.022  2.022  2.022  2.022;
      1.1945 1.1945 1.1945 1.1945 1.1945 1.1945 1.1945 1.1945 1.1945;
       .36   .36   .36   .36   .36   .36   .36   .36   .36;
       .295  .295  .295  .295  .295  .295  .295  .295  .295;
       .265  .265  .265  .265  .265  .265  .265  .265  .265;
       .25   .25   .25   .25   .25   .25   .25   .25   .25;
       .235  .235  .235  .235  .235  .235  .235  .235  .235;
       .185  .185  .185  .185  .185  .185  .185  .185  .185;
       .14   .14   .14   .14   .14   .14   .14   .14   .14;
       .11   .11   .11   .11   .11   .11   .11   .11   .11;
       .08   .08   .08   .08   .08   .08   .08   .08   .08;
       .05   .05   .05   .05   .05   .05   .05   .05   .05;
       .03   .03   .03   .03   .03   .03   .03   .03   .03;
       .02   .02   .02   .02   .02   .02   .02   .02   .02];

a=alpha.*180/pi;
for j = 1:length(a)
  if a(j) < -180
      a(j) = a(j) + 360;
  elseif a(j) > 180
      a(j) = a(j) - 360;
  end
end
mach = abs(mach);
CL = interp2(Mach_CL,aoa_CL,cl_tab,mach,a);
CD = interp2(Mach_CD,aoa_CD,cd_tab,mach,a);
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX M. VR12CLCD.M

```
function [CL,CD]=vr12clcd(alpha)
% vr12clcd calculates CL and CD for the VR-12 airfoil
% given angle of attack (alpha) in radians
% [CL,CD]=vr12clcd(alpha)
CL=zeros(size(alpha));
CD=zeros(size(alpha));
a=alpha*180/pi;

chk=(a>=20 & a<=180);
CL=CL+chk.*(1.1733-0.018879*a+1.5752e-3*a.^2-3.1925e-5*a.^3+2.0949e-7*a.^4-4.3807e-
10*a.^5);

chk=(a>=-180 & a<=-50);
CL=CL+chk.*(-4.6183-0.1923*a-3.5554e-3*a.^2-3.3273e-5*a.^3-1.4528e-7*a.^4-2.3003e-
10*a.^5);

chk=(a>-50 & a <-30);
CL=CL+chk.*(-0.22114+0.020857*a+2.8571e-4*a.^2);

chk=(a>=-30 & a <=-10);
CL=CL+chk.*(-1.11-0.12383*a-0.01515*a.^2-6.8667e-4*a.^3-1e-5*a.^4);

chk=(a<20 & a>-10);
CL=CL+chk.*(0.11976+0.12341*a+5.5841e-4*a.^2-2.0652e-4*a.^3);

chk=(a>=17 & a<=180);
CD=CD+chk.*(-0.26376+0.017917*a+6.9927e-4*a.^2-9.1137e-6*a.^3+2.6277e-8*a.^4);

chk=(a>=-180 & a<=-10);
CD=CD+chk.*(-0.17486-0.034463*a-1.0233e-4*a.^2-2.8958e-6*a.^3-4.6577e-8*a.^4-1.5557e-
10*a.^5);

chk=(a>-10 & a<=0);
CD=CD+chk.*(9.8678e-3+3.4934e-3*a+1.4844e-3*a.^2-1.3564e-4*a.^3-1.0936e-5*a.^4);

chk=(a>0 & a<=15);
CD=CD+chk.*(9.8e-3+7.0457e-4*a+5.6104e-5*a.^2-4.1151e-5*a.^3+3.8695e-6*a.^4);

chk=(a>15 & a<17);
CD=CD+chk.*(-1.33+1.325e-1*a-2.5e-3*a.^2);
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Arcidiacono, P. J., “Theoretical Performance of Helicopters Having Second and Higher Harmonic Feathering Control,” Journal of the American Helicopter Society, Vol. 6, April 1961.

Ashby, D., Eadie, W. and Montoro, G. J., “An Investigation of the Reverse Velocity Rotor Concept and Its Application to High Speed Rotocraft,” AIAA Powered Lift Conference, November 2002.

Eccles, D. M., “A Validation of the Joint Army/Navy Rotorcraft Analysis and Design Software by Comparison with H-34 and UH-60A Flight Test,” Department of Aeronautics and Astronautics, Naval Postgraduate School, Monterey, California, December 1995.

Ewans, J. R. and Krauss, T. A., “Model Wind Tunnel Test of a Reverse Velocity Rotor System,” Fairchild Report HC144R1070, January 1973.

Ewans, J. R., McHugh, F. J., Seagrist, R. P. and Taylor, R. B., “Further Model Wind Tunnel Test of a Reverse Velocity Rotor System,” Fairchild Report HC171R1089, July, 1975.

Gerstenberger, W. and Wood, E. R., “Analysis of Helicopter Aeroelastic Characteristics in High-Speed Flight, AIAA Journal,” November 1963.

Ham, N. D., “Helicopter Individual Blade Control and Its Applications,” Proceedings of the 39th Forum of the American Helicopter Society, May 1983.

Koch, A., “Ship-to-Objective Lift Still a Key Issue, Says US Study,” Jane’s Defence Weekly, Vol. 40, Issue 8, August 27, 2003.

Leishman, J. G., “Principles of Helicopter Aerodynamics,” Cambridge University Press, 2000.

Nicholson, R. K., “Computer Code for Interactive Preliminary Design Using a Harmonic Balance Method for Rotor Trim,” Department of Aeronautics and Astronautics, Naval Postgraduate School, Monterey, California, September 1993.

Prouty, Raymond W., “Helicopter Performance, Stability, and Control,” 1990.

Wood, E. R., Aaron, R. and Gutierrez, J., “Joint Heavy Lift Expeditionary Warfare Aircraft Solution,” Presented at the American Helicopter Society 59th Annual Forum, Phoenix, Arizona, May 6-8, 2003.

Wood, E. R. and Buffalano, A. C., “Parametric Investigation of the Aerodynamic and Aeroelastic Characteristics of Articulated and Rigid (Hingeless) Helicopter Rotor Systems,” TRECOM Technical Report 64-15, April 1964.

Wood, E. R., Powers, R. W., Cline, J. H., and Hammond, E. C., “On Developing and Flight Testing a Higher Harmonic Control System,” Journal of the American Helicopter Society, January 1985.

Wood, E. R. and Van Riper, S. G., “Reverse Velocity Rotor Approach for Design of a STOVL Heavy Lift Transport for Expeditionary Warfare,” Presented at the American Helicopter Society 59th Annual Forum, Phoenix, Arizona, May 6-8, 2003.

Zientek, T., “Expanding the Flight Envelope with a Segmented Rotor,” 57th Annual AHS Forum, May 2001.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. AFIT/LD AFIT Academic Library
   Air Force Institute of Technology
   Dayton, Ohio

4. Chairman
   Department of Mechanical and Astronautical Engineering
   Naval Postgraduate School
   Monterey, California